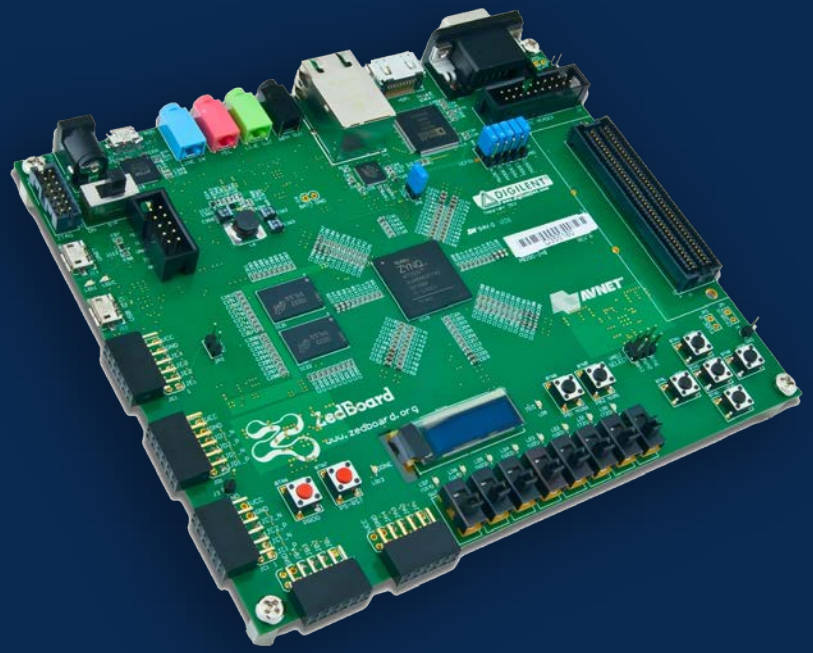
TECHNISCHE UNIVERSITÄT DRESDEN

Informatik » Institut für Technische Informatik » Professur VLSI-Entwurfssysteme, Diagnostik und Architektur

Implementierung eines Linux-Gerätetreibers zur dynamischen Allokation von isolierten Kommunikationskanälen zu partiell konfigurierten FPGA-Kernen in einem Zynq-System

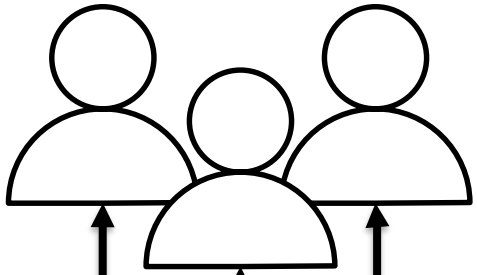
Großer Beleg



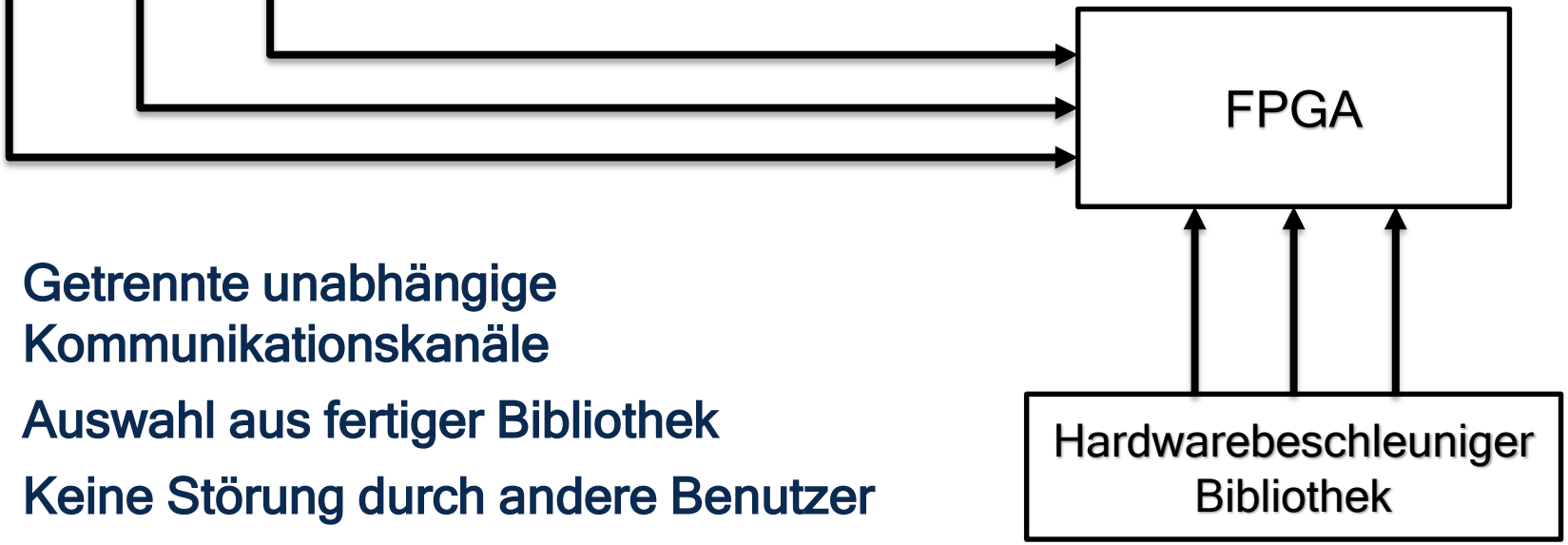
Björn Gottschall
Dresden, 16.02.2017

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Inhaltsverzeichnis



- Flexibilität des FPGAs in Verbindung mit einem Mehrbenutzerbetriebssystem
- Initialisierung und Freigabe von Hardwarebeschleunigern
- Aufteilung und effiziente Nutzung von Ressourcen



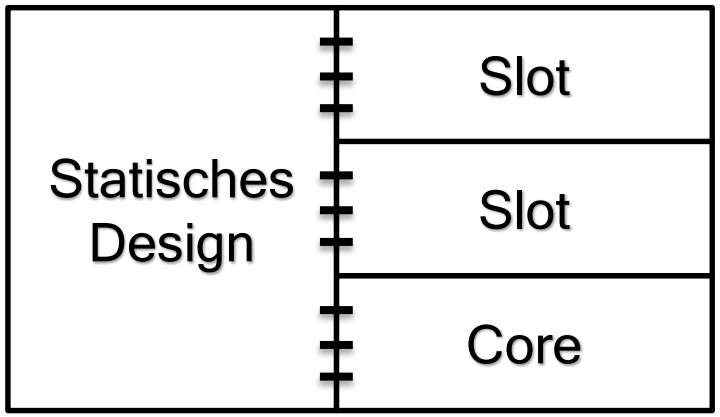
- Getrennte unabhängige Kommunikationskanäle
- Auswahl aus fertiger Bibliothek
- Keine Störung durch andere Benutzer

Partielle Rekonfiguration

- Aufteilung in ein statisches Design und partiell rekonfigurierbare Fenster (Slots)

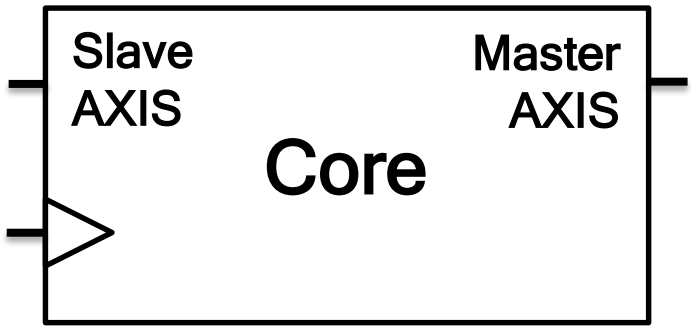
Statisches Design:

- Wird einmalig konfiguriert
- Enthält Schnittstellen zwischen Processing System und den Cores als auch deren Entkoppellogik



- Slot
  kleinste sinnvolle Konfiguration eines Fensters (frei)
- Core
  Hardwarekern zur Verwendung durch Benutzer

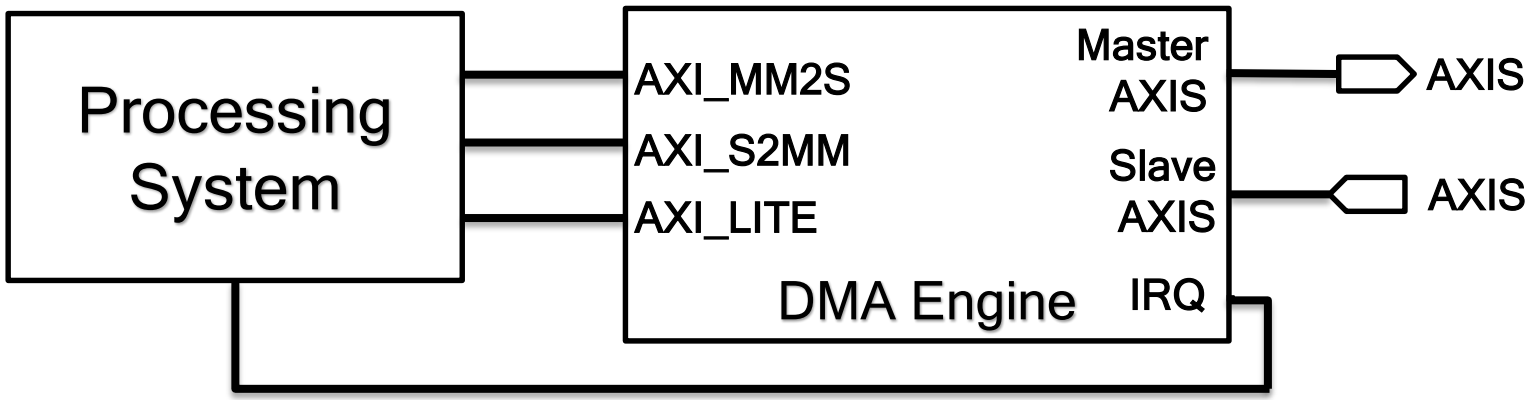
- Jeder Slot besitzt ein Takt und eine AXIS Slave- und Master-Schnittstelle
- Besondere Aufmerksamkeit auf TREADY legen, da die Senke nicht durchgehend bereit ist
- Zusicherung ganzer Übertragungen macht TKEEP überflüssig
- Datenbreite variabel aber abhängig vom statischen Design und für jeden Core gleich



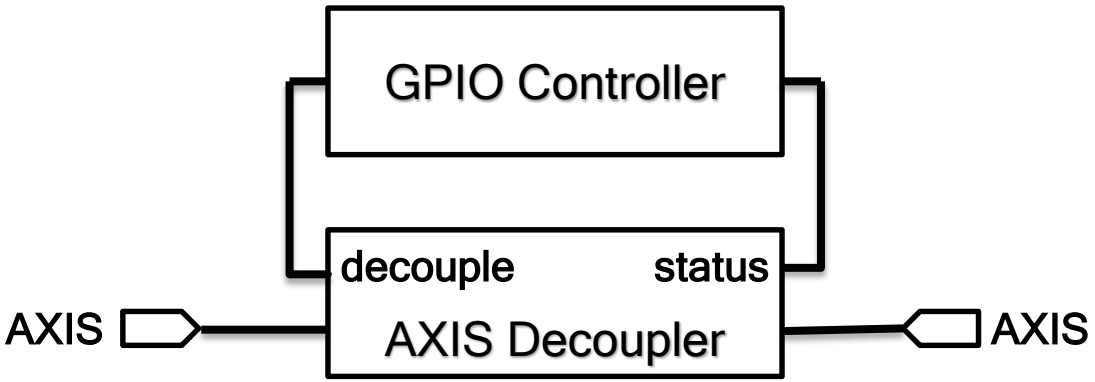
AXI Stream Interface Signale:

- TDATA Daten
- TVALID Gültigkeit der Daten
- TLAST Markiert letzte Übertragung
- TREADY Bereit zu Empfangen

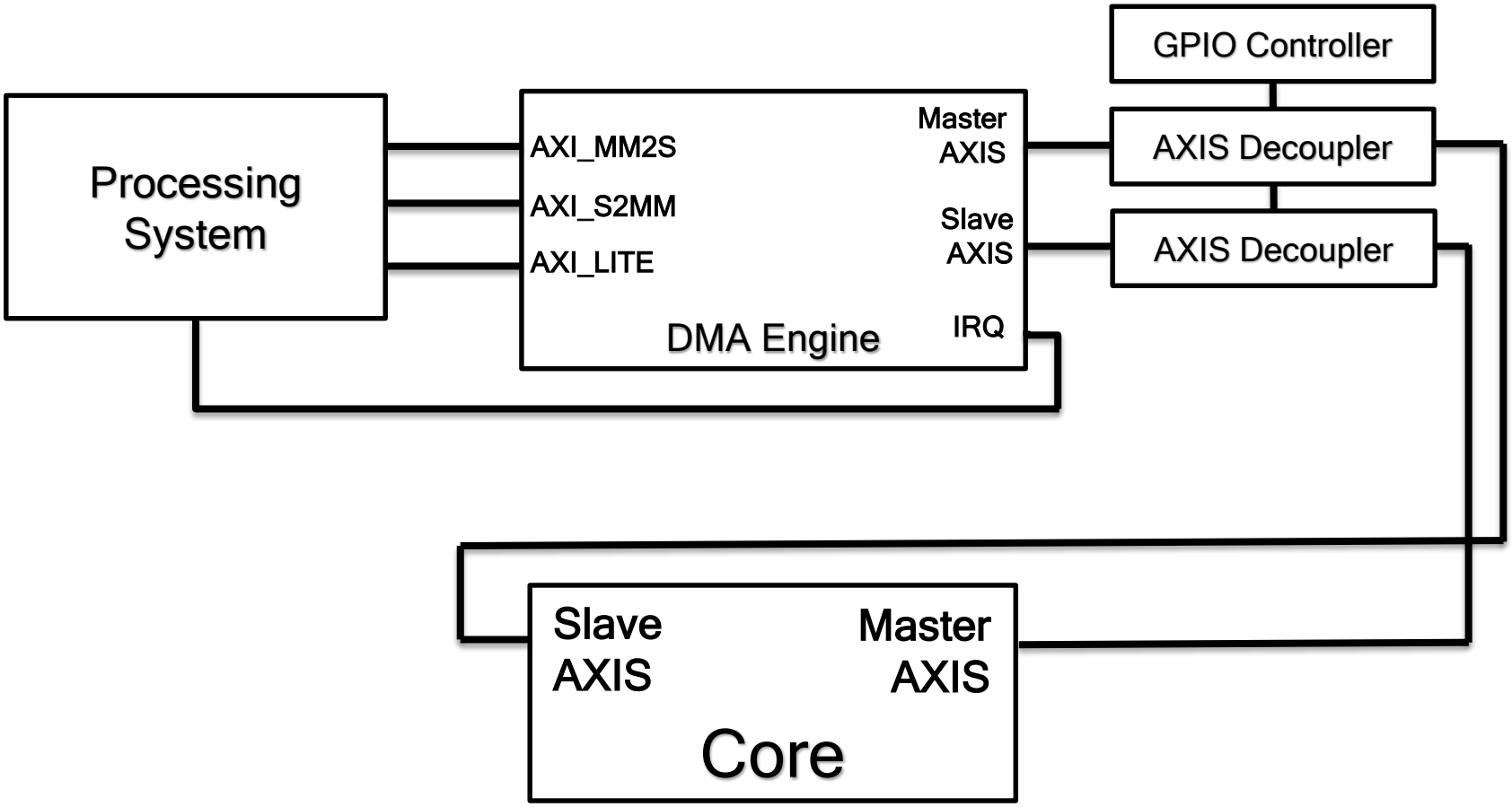
- Jeder Slot ist mit einer Xilinx DMA Engine verbunden (AXIS)
- Eine AXI Lite Schnittstelle zum Konfigurieren der Engine über den General Purpose Port
- Zwei AXI Schnittstellen zum Lesen und Schreiben im RAM verbunden mit einem High Performance Port
- Ein Interruptvektor für die Rückmeldung an den Prozessor



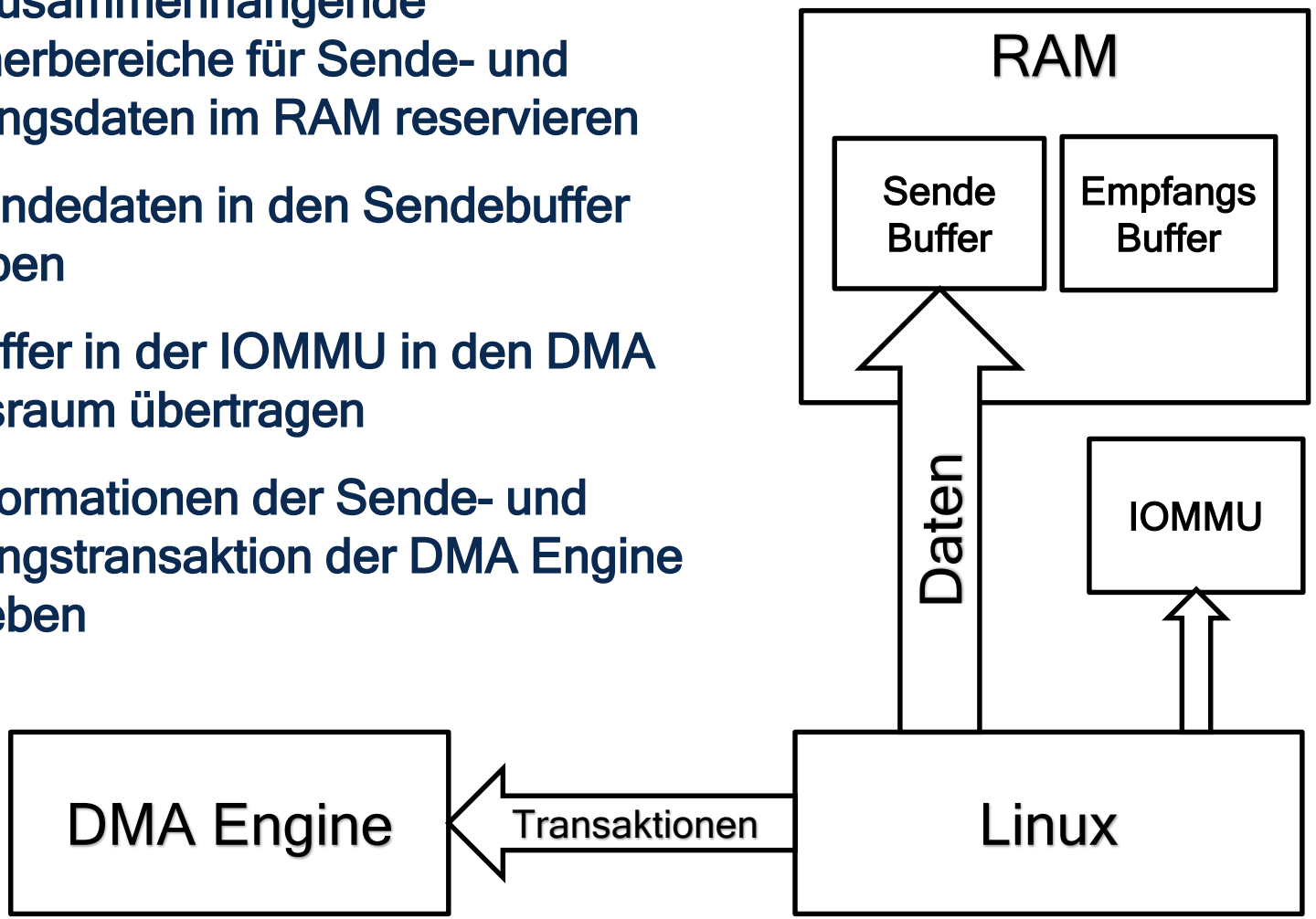
- Während einer partiellen Rekonfiguration können undefinierte Signale an Schnittstellen anliegen
- Jede AXIS Schnittstelle wird mit einem Entkoppler von der DMA Engine getrennt
- Über GPIO Pins kann „decouple“ gesetzt werden und „status“ ausgelesen werden



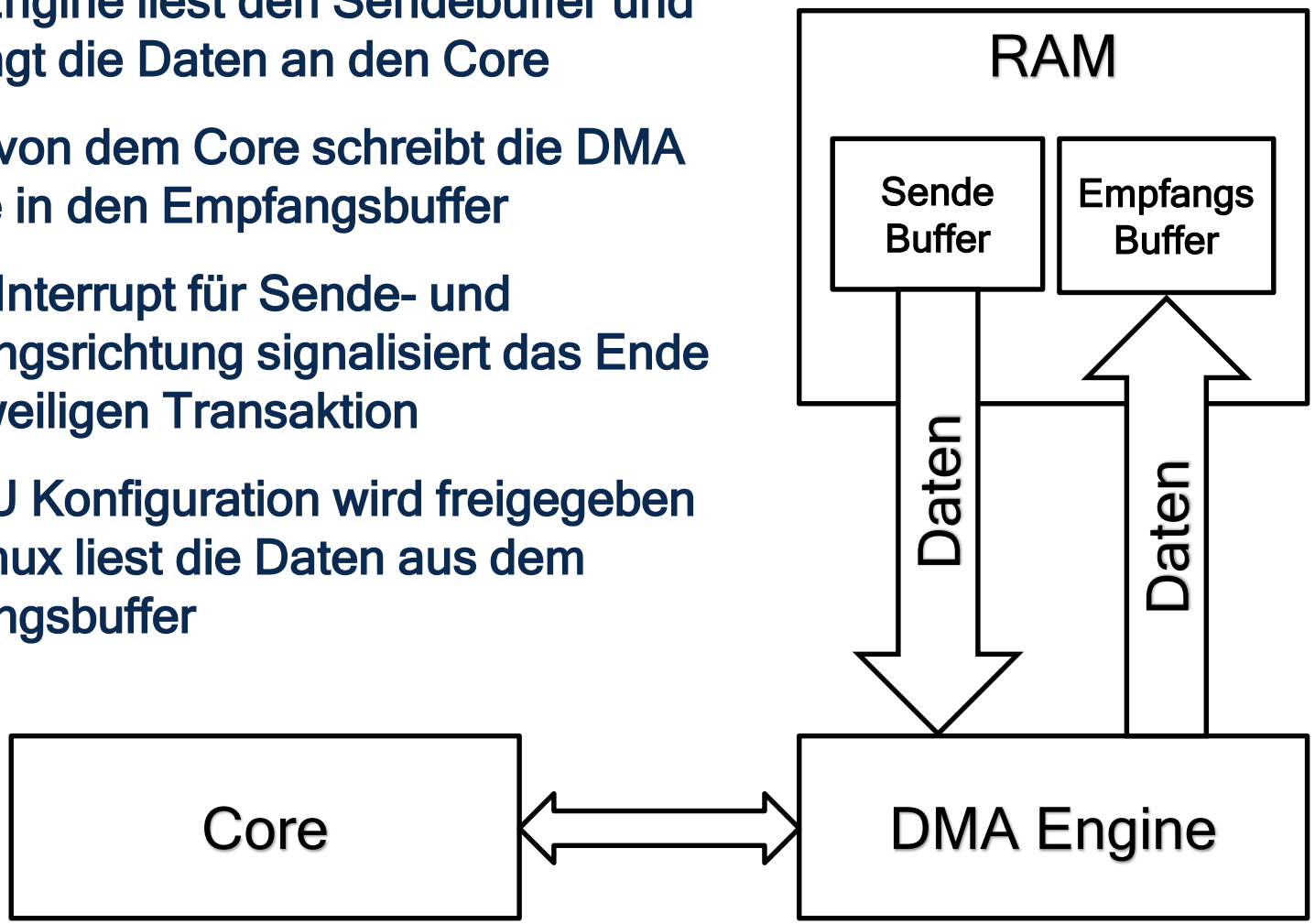
FPGA - Überblick



1. Zwei zusammenhängende Speicherbereiche für Sende- und Empfangsdaten im RAM reservieren
2. Die Sendedaten in den Sendebuffer schreiben
3. Die Buffer in der IOMMU in den DMA Adressraum übertragen
4. Die Informationen der Sende- und Empfangstransaktion der DMA Engine übergeben

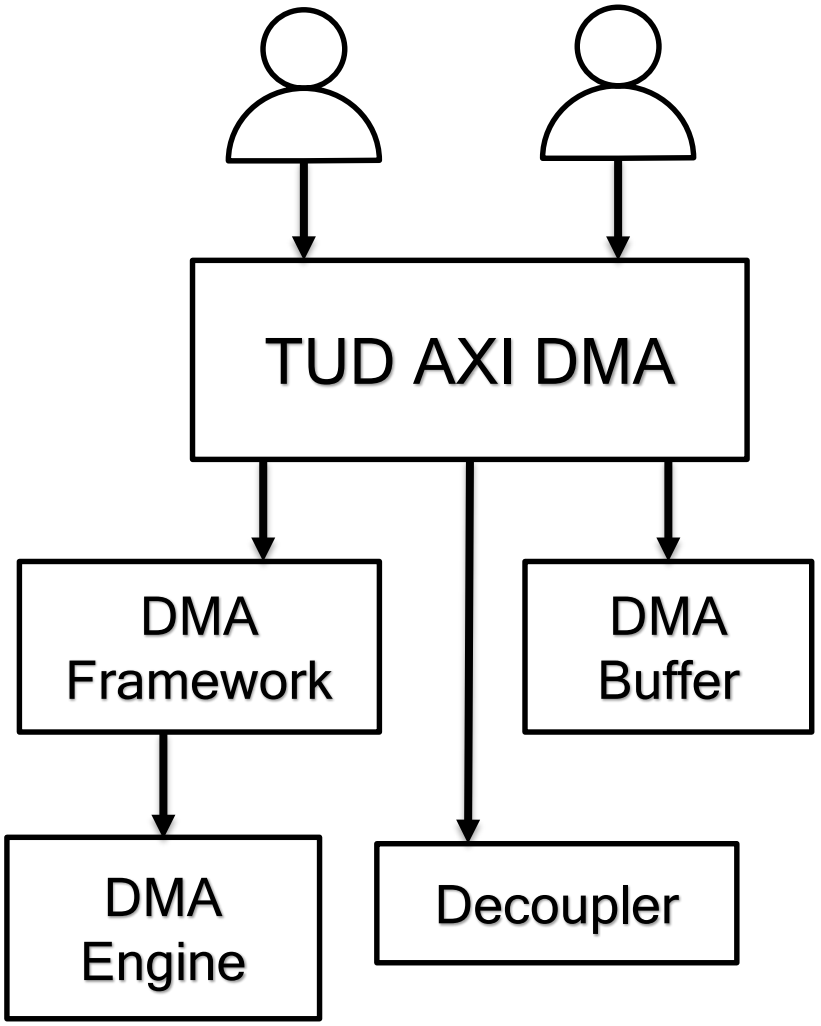


5. DMA Engine liest den Sendebuffer und überträgt die Daten an den Core
6. Daten von dem Core schreibt die DMA Engine in den Empfangsbuffer
7. Je ein Interrupt für Sende- und Empfangsrichtung signalisiert das Ende der jeweiligen Transaktion
8. IOMMU Konfiguration wird freigegeben und Linux liest die Daten aus dem Empfangsbuffer



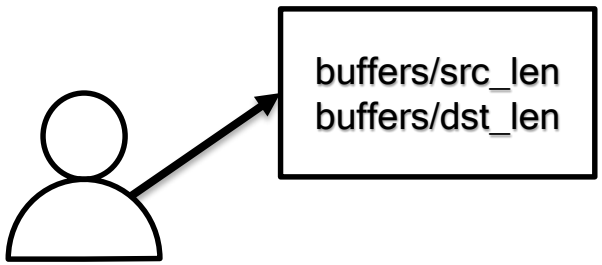
Aufgaben

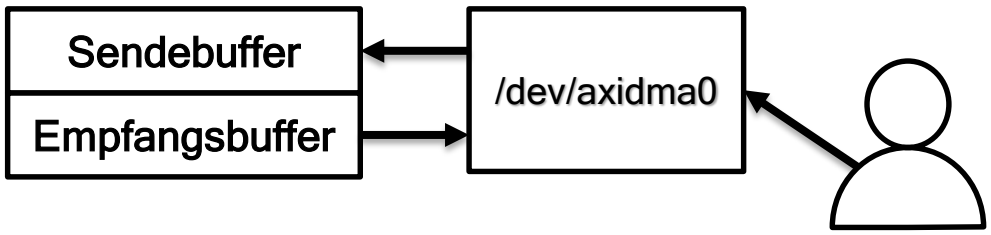
- Verwaltung der DMA Buffer
- Ausführung der DMA Transaktionen mittels DMA Framework
- Entkoppeln der Schnittstellen für die partielle Rekonfiguration
- Bereitstellen einer Informations- und Konfigurationsschnittstelle
- Schreiben und Lesen der DMA Buffer



DMA Buffer

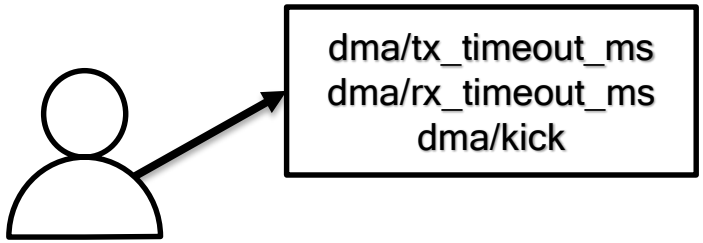
- Bei Geräteinitialisierung werden zwei zusammenhängende Speicherbereiche reserviert
- Verwendung des Contigous Memory Allocators (CMA) für Speicher größer 4 MiB
- Benutzer kann verwendete Größe selbst bestimmen (vielfache Datenbreite, mindestens Datenbreite, maximal 2^{23} - 1 Byte)
- Schreiben und Lesen mittels eines Character Device





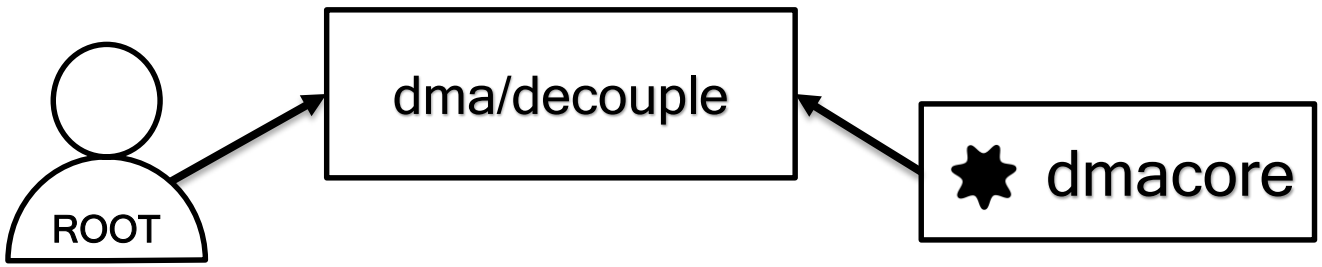
DMA Framework

- Xilinx DMA Treiber bindet die DMA Engine an das Linux DMA Framework an
- Reservierung von spezifischen DMA Channels durch Devicetree-Referenzen und Vorbereitung der Transaktion mit gewählten Parametern
- Abbilden der DMA Buffer in den Adressraum der IOMMU
- Übertragen und Auslösen der DMA Transaktionen
- Warten auf Interrupt oder Timeout

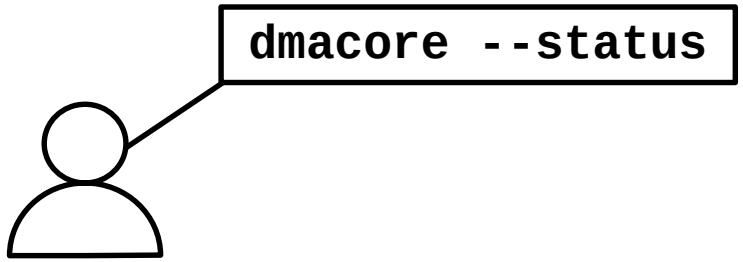


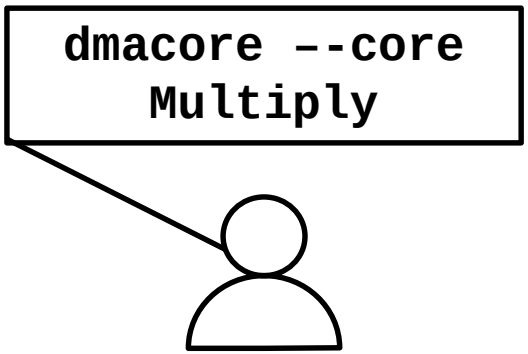
Entkoppeln

- Xilinx GPIO Treiber bindet die GPIO Pins an die Linux GPIO Schnittstelle an
- Bei Geräteregistrierung werden zwei GPIO Pins initialisiert
- Lesen von „decouple“ gibt den Wert des Status GPIO Pins zurück
- Schreiben von „decouple“ setzt den Wert des decouple GPIO Pins
- Vorgang ist unabhängig von der DMA Transaktion

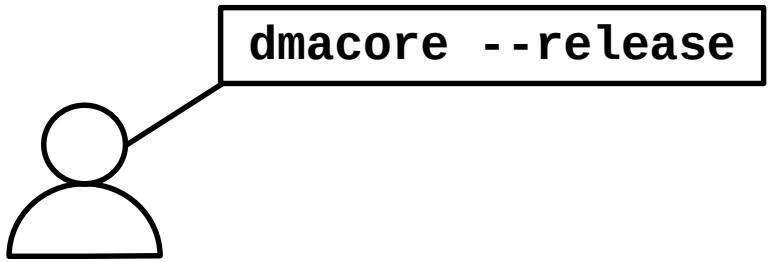


- Verwaltung der Slots und Cores
- Auflistung vorhandener Cores mit Beschreibung





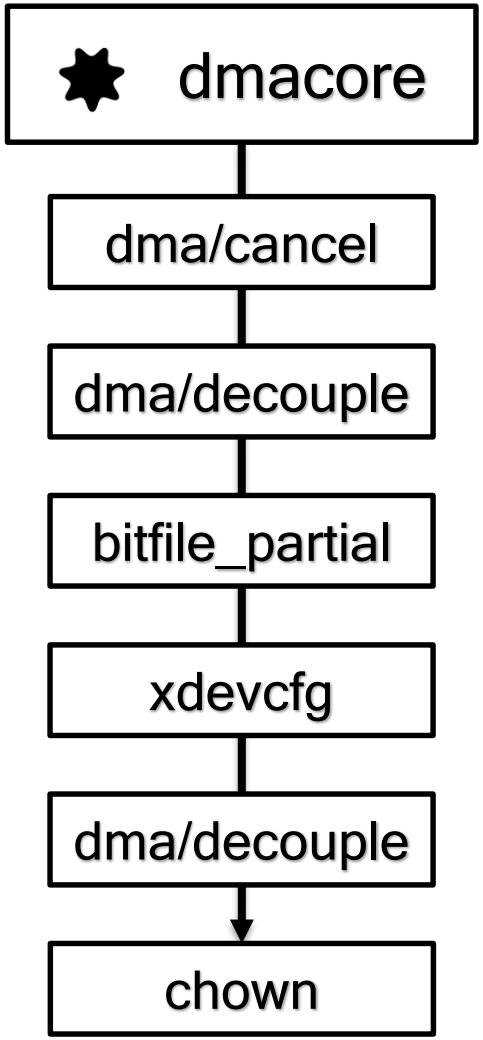
- Anzeige von eigenen und freien Slots
- Ermöglicht unprivilegierten Benutzern die Initialisierung eines Cores



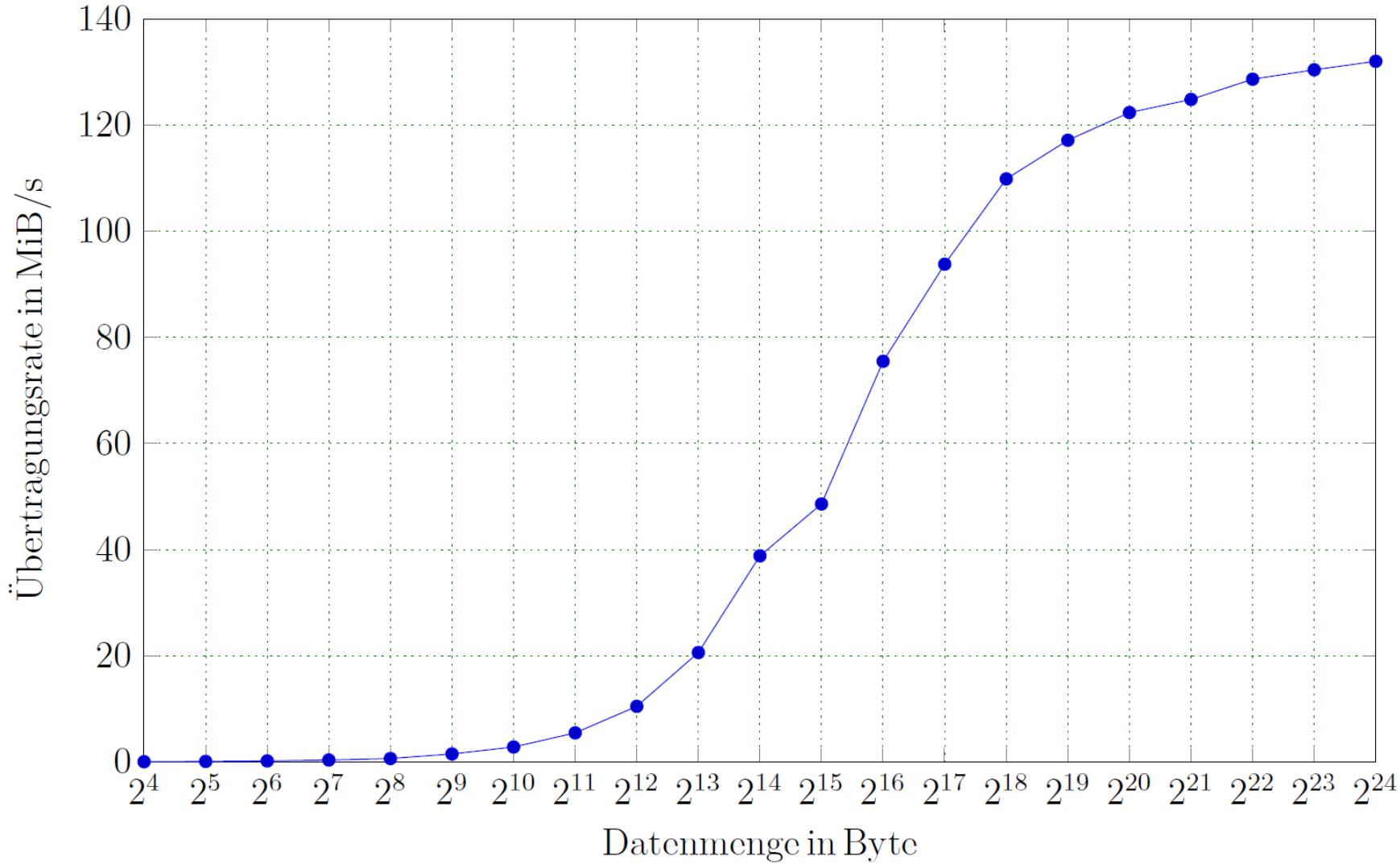
- Applikationsschnittstelle
- Einfaches Bash-Skript

Rekonfiguration eines Slots

- DMA Transaktionen werden vor dem Rekonfigurieren eines Slots beendet
- Entkopplung der Schnittstellen des Slots
- Partielle Rekonfiguration mit dem gewünschten Bitstream
- Kopplung der Schnittstellen des Slots
- Übergabe der Rechte der Treiberschnittstellen an den Benutzer



Fazit - Performance



- ✓ Aufteilung von FPGA Ressourcen mittels Hardwarekernen
- ✓ Störungsfreie Initialisierung, Verwendung und Freigabe von Hardwarekernen von normalen Benutzern
- ✓ Isolierte unabhängige Kommunikationskanäle
- ✓ Gute Performance ohne Optimierungen
- o Treiber und FPGA-Design Optimierungen
- o Erweiterung des Treibers
  - – Software Interrupts
  - – Flexiblere DMA Transaktionen
  - – Error Reporting
  - – Driver Memory to Userpace Mapping

Praktisches Beispiel

Vielen Dank!

- [1] Xilinx UG909: Partial Reconfiguration (Vivado)
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_2/ug909-vivado-partial-reconfiguration.pdf
- [2] kernel.org: DMA API
  https://www.kernel.org/doc/Documentation/DMA-API-HOWTO.txt
- [3] kernel.org: DMA Slave API
  https://www.kernel.org/doc/Documentation/dmaengine/client. txt
- [4] John Linn: Xilinx Präsentation zur Verwendung der DMA API aus Linux heraus
  https://forums.xilinx.com/xlnx/attachments/xlnx/ELINUX/10658/1/drivers-session4-dma-4public.pdf

Multiplikationen	Daten in Bytes	Zeit in µS	Bandbreite in MiB/s
1	16	349	0,04
2	32	318	0,09
4	64	338	0,18
8	128	337	0,36
16	256	379	0,64
32	512	326	1,5
64	1024	347	2,81

Multiplikationen	Daten in KiB	Zeit in µS	Bandbreite in MiB/s
128	2	357	5,47
256	4	373	10,47
512	8	379	20,61
1024	16	462	38,82
2048	32	643	48,6
4096	64	828	75,48
8192	128	1333	93,77
16384	256	2276	109,84
32768	512	4269	117,12
65536	1024	8174	122,34

Multiplikationen	Daten in MiB	Zeit in µS	Bandbreite in MiB/s
131072	2	16024	124,81
262144	4	31096	128,63
524288	8	61355	130,39
1048575	16 MiB - 16 Byte	121210	132