TECHNISCHE UNIVERSITÄT DRESDEN

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

Konzeption heterogener Mehrkernprozessoren für die HD-Videodekodierung auf Basis der SoC-Plattform „LEON3“

Verteidigung zum Großen Beleg

Sebastian Heinze

Sebastian.Heinze1@mailbox.tu-dresden.de

Dresden, 06.11.2013

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Gliederung

1. Aufgabenstellung
2. Literaturrecherche
3. Analyse
4. Entwurf
5. Zusammenfassung, Wertung und Ausblick
6. Literatur

1. Aufgabenstellung

Motivation:

- Dekodierung von HD Videomaterial sehr zeitaufwändig in Software, auch auf aktuellen Prozessoren
- Beschleunigung der Dekodierung auf SoCs durch zusätzliche Kerne

1. Aufgabenstellung

Zielstellung:

- Entwicklung geeigneter Konzepte zur hardwarebeschleunigten Videodekodierung eines Theora-codierten Videos auf der Leon3 SoC Plattform
- Bewertung der entworfenen Konzepte

1. Aufgabenstellung

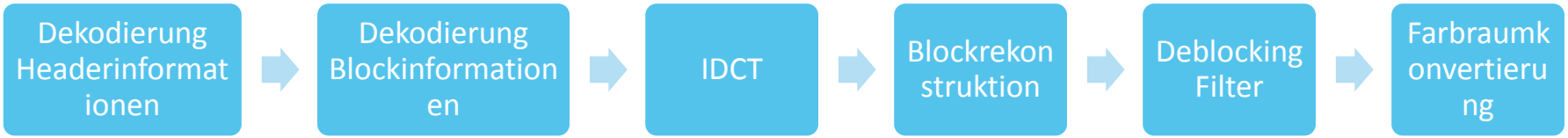
Durchgeführte Arbeitsschritte:

- Literaturrecherche
- Implementierung & Inbetriebnahme des Leon3 SoC Systems auf einem Virtex5 FPGA-Board
- Laufzeitanalyse der Theora Videodekodierung anhand unterschiedlicher HD-Testvideos
- Analyse der Ergebnisse der Laufzeitanalyse im Hinblick auf Leistungsfähigkeit & Kosten einer Implementierung bestimmter Funktionen
- Zusammenfassung der Analyseergebnisse in einem Entwurf

2. Literaturrecherche

Theora Codec: [Th13]

- Ursprünglich von On2 Technologies als VP3 entwickelt (Teil der TrueMotion Videocodec-Familie)
- Breite Unterstützung unter Software
- Sehr geringe Anzahl an Theora unterstützender Hardware (schwerer Stand gegen h.264)
- Implementierung ist bereits auf Hardwarebeschleunigung ausgelegt



[1]

2. Literaturrecherche

Hardwareimplementierung v. Theora auf FPGA: (letzter Stand von 2007) [Cst07]

Testvideo von Costa

- 96x80 Pixel
- 15 Sekunden
- Szene eines Fußballspielers, stationäre Kameraführung
- Ausführung mit Linux ergibt 75s Dekodierungsdauer

2. Literaturrecherche

Hardwareimplementierungen wichtiger Dekoderfunktionen:

Zusammenfassung

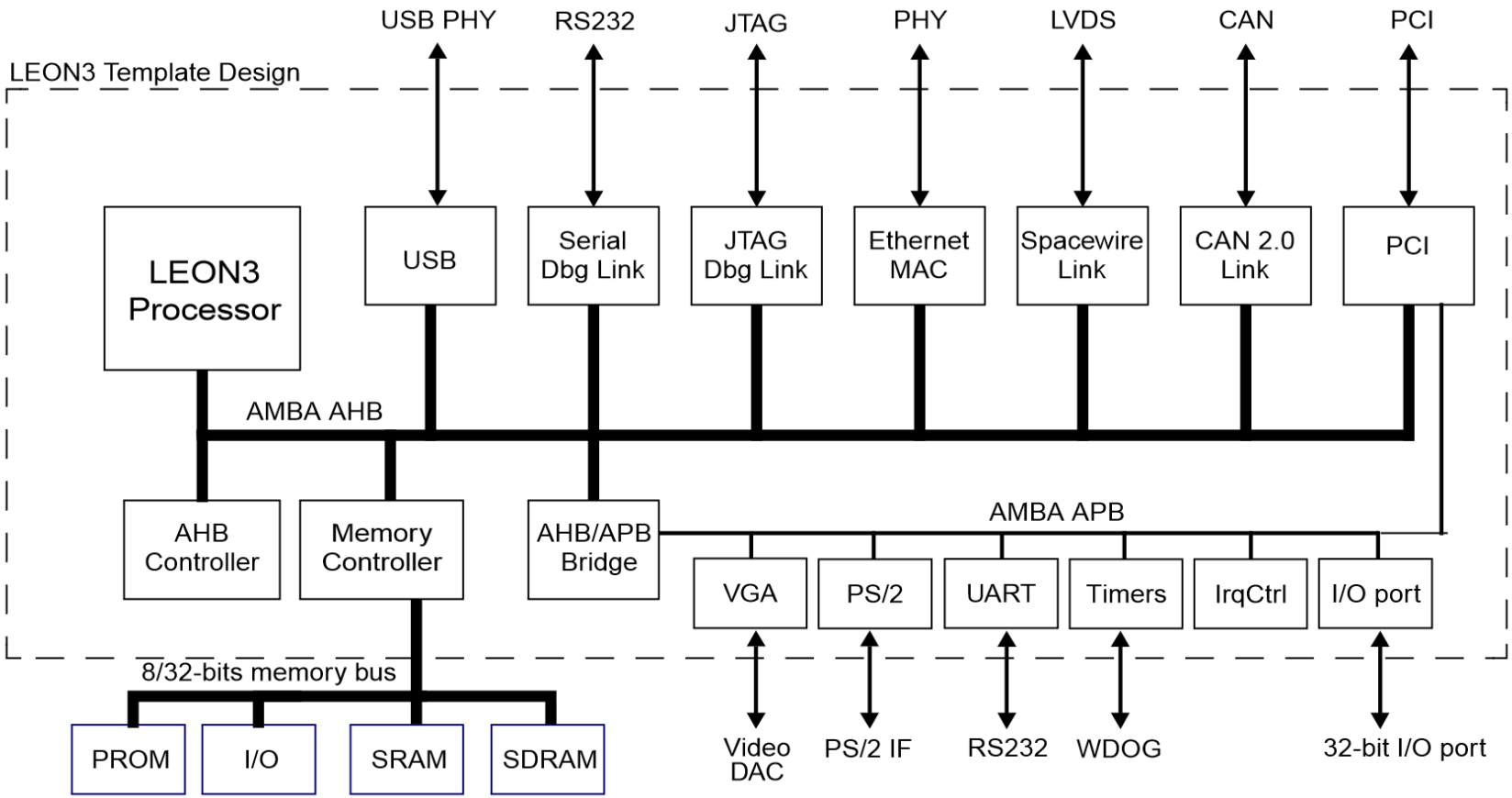
Implementierung	Funktion	Benötigte Takte/Einheit
Correa, 2009 [Cor09]	Deblockingfilter	53
Nadeem, 2010 [Nad10]	Deblockingfilter	192
Swamy, 2005 [Swa05]	IDCT	30 + 8
Aspar, 2000 [Asp00]	Huffman Decoder	1

3. Analyse

Umgebung für die Laufzeitanalyse

- Xilinx Virtex 5 ML505 Board
- Für das ML505 Board angepasstes Leon3 SoC-Design
- Für Leon3 angepasstes Snapgear-Linux
- Mit „–g“ gcc-Option kompilierte Version der dump_video Funktion
- NFS Dateisystem zur Übertragung der Profilingergebnisse

3. Analyse


USB PHY
RS232
JTAG
PHY
LVDS
CAN
PCI
LEON3 Template Design
LEON3 Processor
USB
Serial Dbg Link
JTAG Dbg Link
Ethernet MAC
Spacewire Link
CAN 2.0 Link
PCI
AMBA AHB
AHB Controller
Memory Controller
AHB/APB Bridge
AMBA APB
VGA
PS/2
UART
Timers
IrqCtrl
I/O port
8/32-bits memory bus
PROM
I/O
SRAM
SDRAM
Video DAC
PS/2 IF
RS232
WDOG
32-bit I/O port

[2]

3. Analyse

Laufzeitanalyse:

Testvideos

Video	Auflösung	Dauer (s)	Anzahl der Frames
Ducks720	1280x720	10	250
Bunny720	1280x720	10	241
Bunny1080	1920x1080	10	244

3. Analyse

Laufzeitanalyse:

Ergebnisse (Bsp.: Ducks720)

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
17.52	66.41	66.41	250	0.27	1.44	th_decode_packetin
17.45	132.59	66.18	4977941	0.00	0.00	oc_frag_recon_inter2_c
8.32	164.16	31.56	6779653	0.00	0.00	loop_filter_h
8.27	195.52	31.36	14984144	0.00	0.00	idct8
7.77	224.97	29.46	5017388	0.00	0.00	oc_frag_copy_c
6.63	250.11	25.13	6830934	0.00	0.00	loop_filter_v
4.90	268.67	18.56	5782612	0.00	0.00	oc_state_frag_recon_c
4.38	285.26	16.59	10625967	0.00	0.00	idct8_4
4.09	300.77	15.51				ogg_stream_packetin
3.54	314.20	13.43	2172084	0.00	0.00	oc_idct8x8_c
3.27	326.58	12.38	10804960	0.00	0.00	oc_huff_token_decode
2.07	334.41	7.83	17250	0.00	0.00	oc_state_loop_filter_frag_rows_c
1.51	340.13	5.72	677218	0.00	0.00	oc_frag_recon_intra_c
1.33	345.17	5.04	5105394	0.00	0.00	oc_state_get_mv_offsets
1.02	349.02	3.85	17250	0.00	0.00	oc_state_frag_copy_list_c
1.01	352.86	3.84	5782612	0.00	0.00	oc_state_frag_recon

3. Analyse

Laufzeitanalyse:

Ausgewählte Funktionen (anhand Ducks720)

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
17.52	66.41	66.41	250	0.27	1.44	th_decode_packetin
17.45	132.59	66.18	4977941	0.00	0.00	oc_frag_recon_inter2_c
8.32	164.16	31.56	6779653	0.00	0.00	loop_filter_h
8.27	195.52	31.36	14984144	0.00	0.00	idct8
7.77	224.97	29.46	5017388	0.00	0.00	oc_frag_copy_c
6.63	250.11	25.13	6830934	0.00	0.00	loop_filter_v
4.90	268.67	18.56	5782612	0.00	0.00	oc_state_frag_recon_c
4.38	285.26	16.59	10625967	0.00	0.00	idct8_4
4.09	300.77	15.51				ogg_stream_packetin
3.54	314.20	13.43	2172084	0.00	0.00	oc_idct8x8_c
3.27	326.58	12.38	10804960	0.00	0.00	oc_huff_token_decode
2.07	334.41	7.83	17250	0.00	0.00	oc_state_loop_filter_frag_rows_c
1.51	340.13	5.72	677218	0.00	0.00	oc_frag_recon_intra_c
1.33	345.17	5.04	5105394	0.00	0.00	oc_state_get_mv_offsets
1.02	349.02	3.85	17250	0.00	0.00	oc_state_frag_copy_list_c
1.01	352.86	3.84	5782612	0.00	0.00	oc_state_frag_recon

3. Analyse

Laufzeitanalyse – Verfeinerte Auswahl:

Analyse des Aufwandes einer möglichen Implementierung

Beispiel: **idct8** – zusammengefasste Profilingergebnisse

Video	Ducks720	Bunny720	Bunny1080
Verweilzeit (s)	29,46	11,84	59,066
Aufrufe	5017388	566960	27785856
Verweilzeit/Call (µs)	5,87	2,09	2,13
Takte/Call	470	168	171

3. Analyse

Laufzeitanalyse – Verfeinerte Auswahl:

Analyse des Aufwandes einer möglichen Implementierung

Beispiel: **idct8** – abgeschätzter Aufwand einer Hardwareimplementierung

Abschnitt	**Aufwand (in Takten)**
Speicherlatenz (Lesen)	11
Übertragungsdauer	1
Rechenzeit	38
Speicherlatenz (Schreiben)	1
Gesamt	**51**

3. Analyse

Laufzeitanalyse – Verfeinerte Auswahl:

Zusammenfassung der Ergebnisse (Werte in Anzahl Takten)

Funktion	Ducks720	Bunny720	Bunny1080	Hardware
Huffman	92	88	83	40
IDCT	470	168	171	51
Debl. Filter	372	-	-	73
Frag. Copy	470	491	480	17
Frag. Recogn. (Inter)	980	918	992	79
Frag. Recogn. (Inter2)	1064	1025	1057	79
Frag. Recogn. (Intra)	676	747	753	79
Get Mv. Offsets	79	79	75	23

3. Analyse

Speicherarchitektur:

Beispielhafte Aufstellung für einen Lesezugriff

Speichertakt: 200 MHZ -> 5ns Periodendauer

Abschnitt	Zeit	Anzahl Takte
PRECHARGE	15 ns	3
ACTIVATE	15 ns	3
READ	15 ns	3
2*Abtastung	10 ns	2
Summe	**55 ns**	**11**

4. Entwurf

Speedup:

Zusammenfassung (anhand Ducks720)

Funktion	Speedup
loop_filter	9,12
huffman_token_decode	2,30
oc_frag_copy_c	27,65
oc_frag_recon_inter2_c	13,47
oc_frag_recon_inter_c	12,41
oc_frag_recon_intra_c	8,56
idct8	5,60
idct8_4+idct8_2+idct8_1	2,33
oc_state_get_mv_offsets	3,43

4. Entwurf

Speedup, bezogen auf die Gesamtlaufzeit:
Berechnung

$$S = \frac{T_{software}}{T_{hardware}} = \frac{1}{(1 - \sum_{i=1}^{9} p_i) + \sum_{j=1}^{9} \frac{p_j}{s_j}}$$

i,j … zu durchlaufende Funktionen

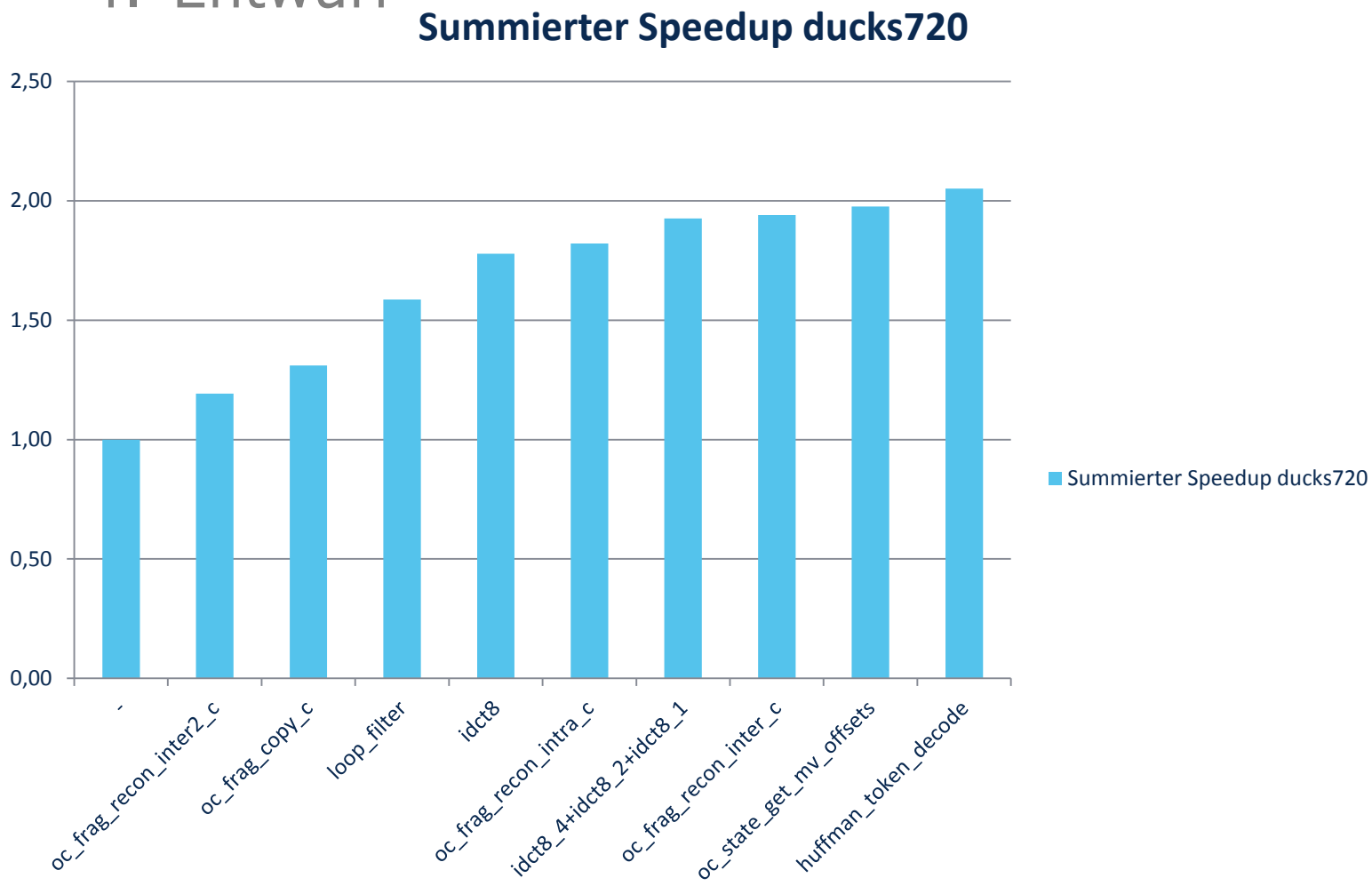
p … Anteil der Funktion an Gesamtlaufzeit der Laufzeitanalyse

s … (ungewichtetes) Speedup der Funktion

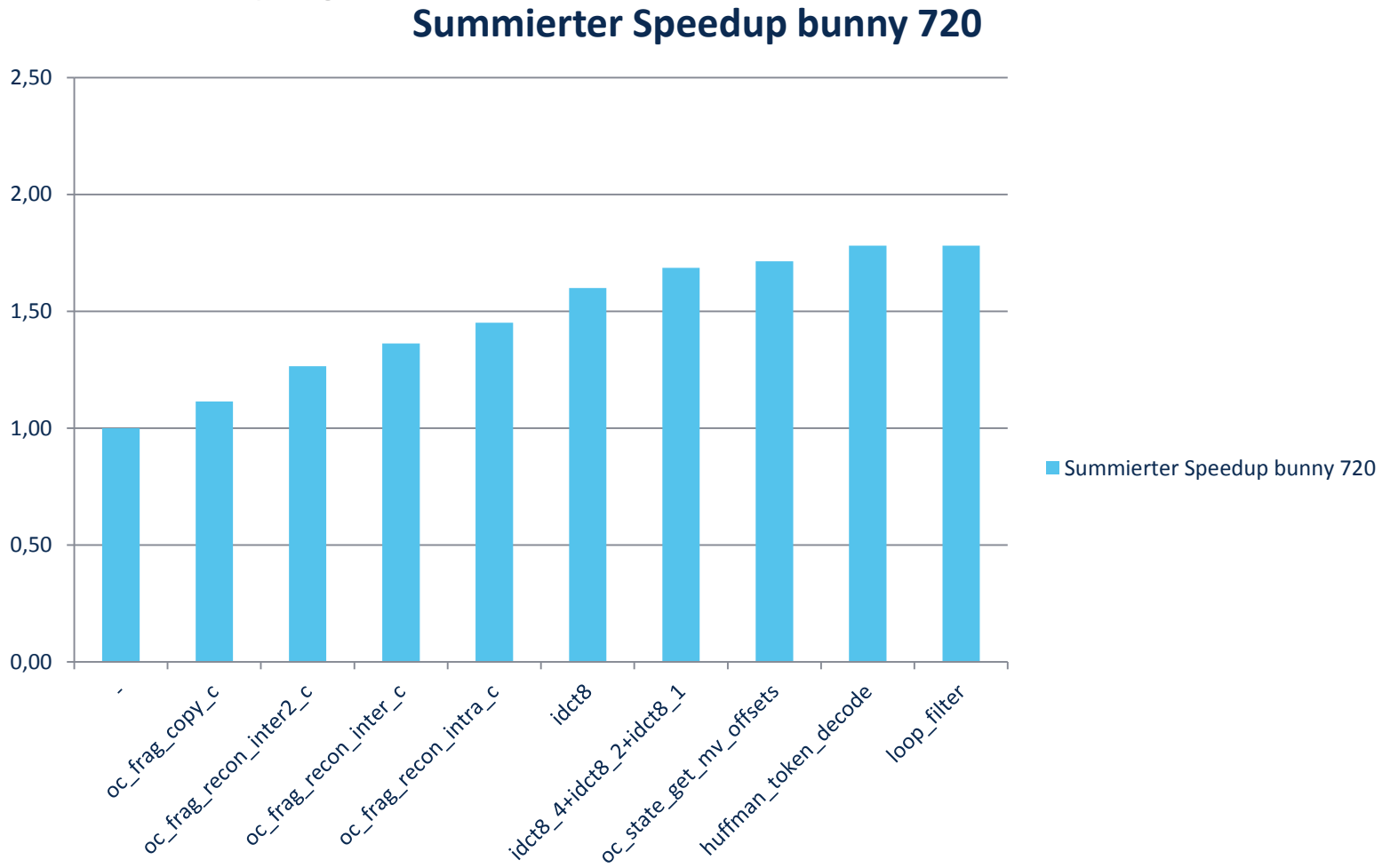
Sortierung der Funktionen anhand des Maßes:

$$Speedup * prozentualer\ Anteil\ an\ Gesamlaufzeit$$

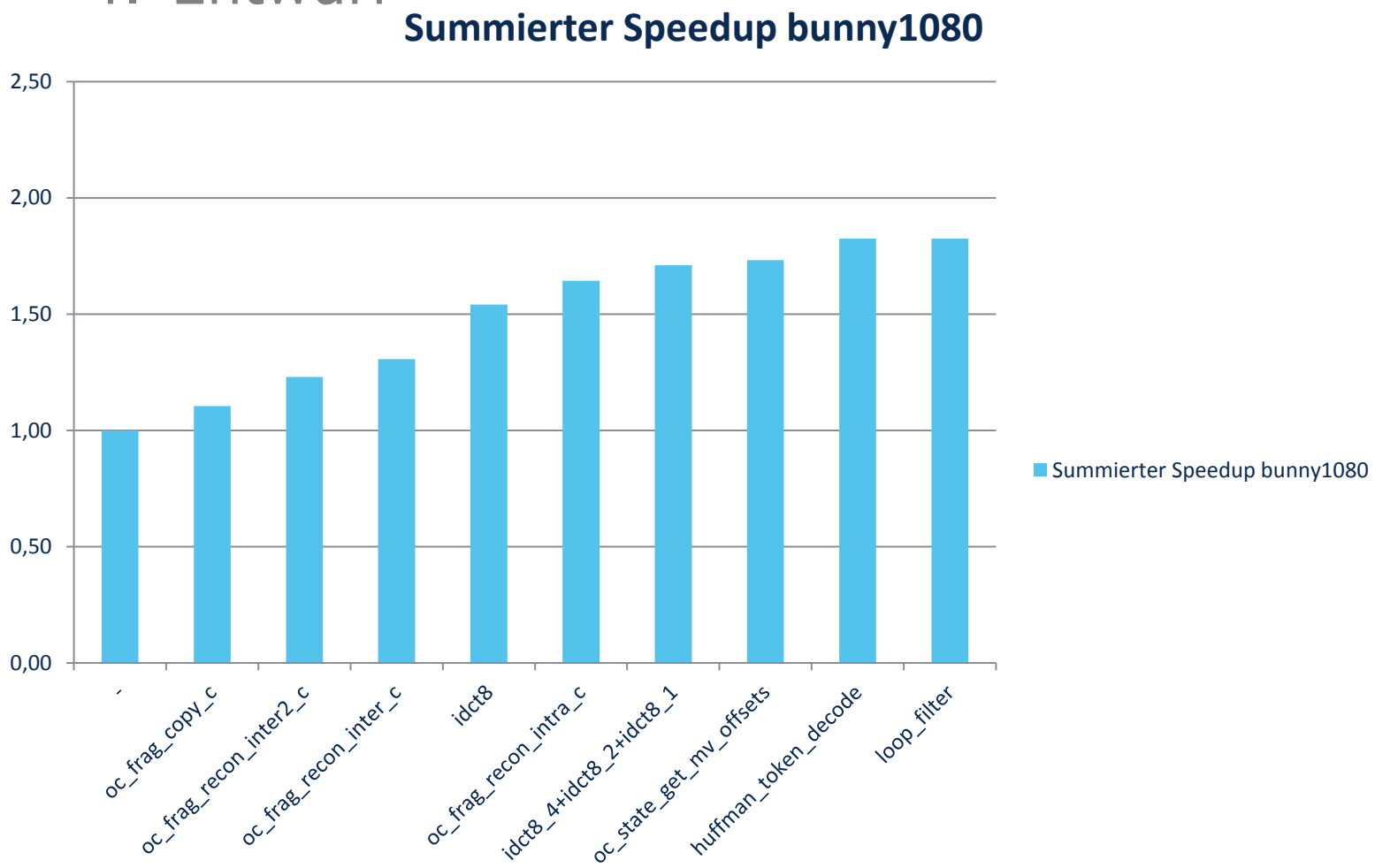
4. Entwurf


Summierter Speedup ducks720
2,50
2,00
1,50
1,00
0,50
0,00
-
oc_frag_recon_inter2_c
oc_frag_copy_c
loop_filter
idct8
oc_frag_recon_intra_c
idct8_4+idct8_2+idct8_1
oc_frag_recon_inter_c
oc_state_get_mv_offsets
huffman_token_decode
Summierter Speedup ducks720

4. Entwurf


Summierter Speedup bunny 720
2,50
2,00
1,50
1,00
0,50
0,00
-
oc_frag_copy_c
oc_frag_recon_inter2_c
oc_frag_recon_inter_c
oc_frag_recon_intra_c
idct8
idct8_4+idct8_2+idct8_1
oc_state_get_mv_offsets
huffman_token_decode
loop_filter
Summierter Speedup bunny 720

4. Entwurf


Summierter Speedup bunny1080
2,50
2,00
1,50
1,00
0,50
0,00
-
oc_frag_copy_c
oc_frag_recon_inter2_c
oc_frag_recon_inter_c
idct8
oc_frag_recon_intra_c
idct8_4+idct8_2+idct8_1
oc_state_get_mv_offsets
huffman_token_decode
loop_filter
Summierter Speedup bunny1080

4. Entwurf

Konzept:

Beschleunigung der Funktionen

- oc_frag_copy_c
- oc_frag_recon_inter2_c
- idct8
- oc_frag_recon_inter_c
- oc_frag_recon_inter_c
- loop_filter
- (idct8_4+idct8_2+idct8_1)

5. Zusammenfassung, Wertung und Ausblick

Zusammenfassung

- Lauffähiges Leon3 SoC auf FPGA Board
- 10 mögliche Funktionen für die Beschleunigung
- Zusammenfassung zu Entwurf
    - Schnelles Speicherinterface
    - Hardwareimplementierung von 6 Funktionen

5. Zusammenfassung, Wertung und Ausblick

Wertung und Ausblick

- Speedup von ca. 2 des vorgestellten Entwurfs bei weitem nicht ausreichend
- Speedup von ca. 100 notwendig, um 25 fps zu erreichen
- Realisierung durch komplette Verarbeitungsblöcke in Hardware denkbar

  →„Makroblock-Pipeline“

6. Literatur

[Th13] Xiph.org. 2013. Theora Main Page. [Online] 2013. [Zitat vom: 03. 05 2013.] http://www.theora.org/

[Cst07] Costa, André Luiz Nazareth da und Terriberry, Timothy B. 2007. Hardware implementation of Theora decoding. Xiph.org. [Online] 2007. [Zitat vom: 14. 05 2013.] http://people.xiph.org/~j/bzr/theora-fpga/doc/leon3_integration/.

[Cor09] Correa, Guilherme, et al. 2009. DESIGN OF AN INTERLAYER DEBLOCKING FILTER ARCHITECTURE FOR H.264/SVC. 2009.

[Nad10] Nadeem, Muhammad, et al. 2010. Low-power, High-throughput Deblocking Filter for H.264/AVC. 2010.

[Swa05] Swamy, Ramkrishna, et al. 2005. A fast, pipelined implementation of a twodimensional Inverse Discrete Cosine Transform. 2005.

[Asp00] Aspar, Zulfaka; Yusof, Zulkalnain Mohd und Suleiman, Ishak. 2000. Parallel Huffman Decoder with an Optimize Look UP TableOption on FPGA. 2000.

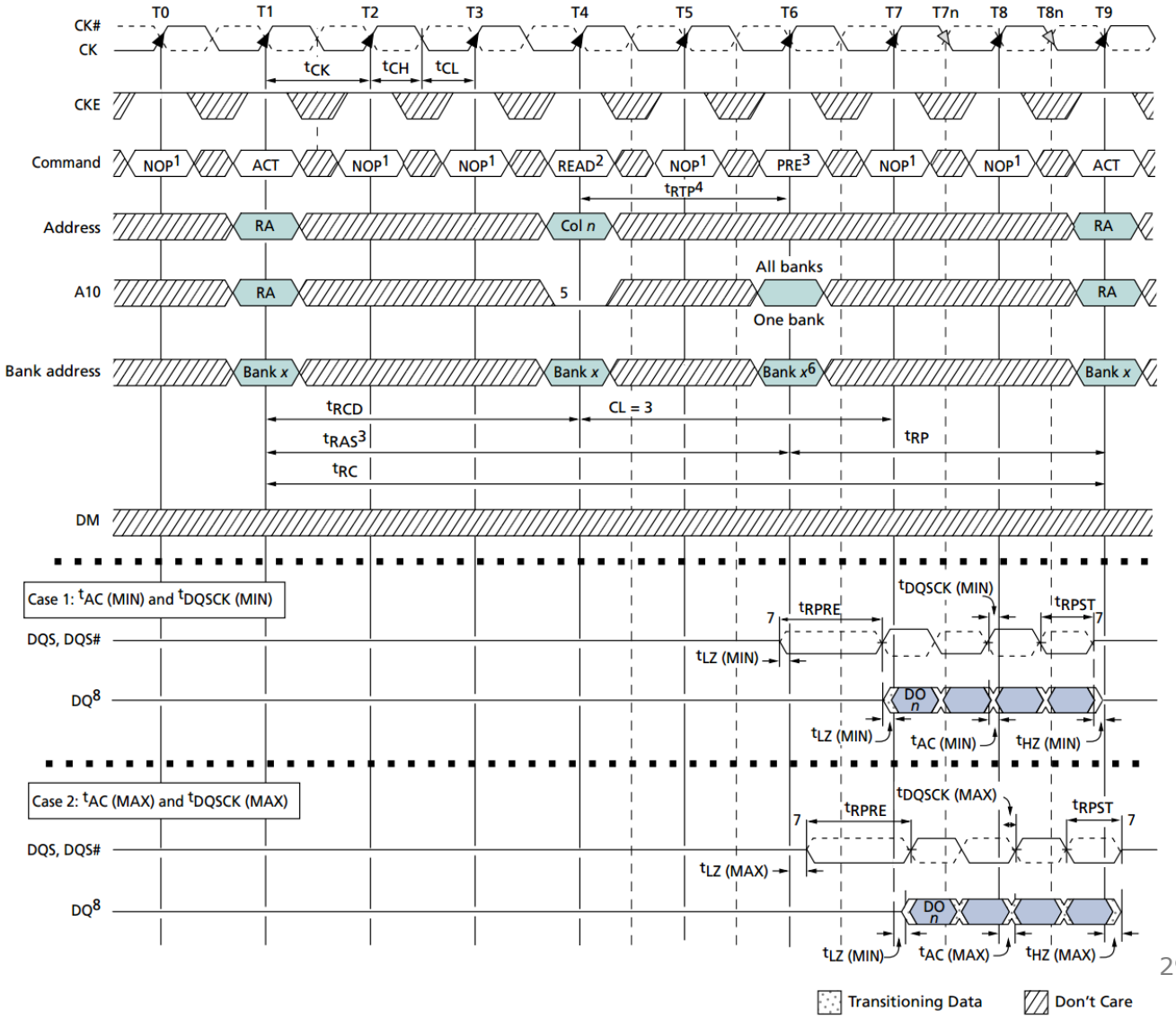
6. Literatur

[1] Basierend auf: Xiph Foundation. 2011. Theora Specification. [Online] 16.03.2011; S.16- S.18
[2] Gaisler Aeroflex. 2013. GRLIB IP Library User's Manual [Online] August 2013; S.6
[3] MICRON. 2004. DDR2 SDRAM. [Datenblatt] s.l. : MICRON, 2004. S.105
[4] MICRON. 2004. DDR2 SDRAM. [Datenblatt] s.l. : MICRON, 2004. S.100

Vielen Dank für ihre Aufmerksamkeit!

Anhang

Speicherzugriff READ, ohne Auto-Precharge [3]



Anhang

Speicherzugriff READ, Consecutive Read Burst