18.341 Projekt

Systementwurf

—

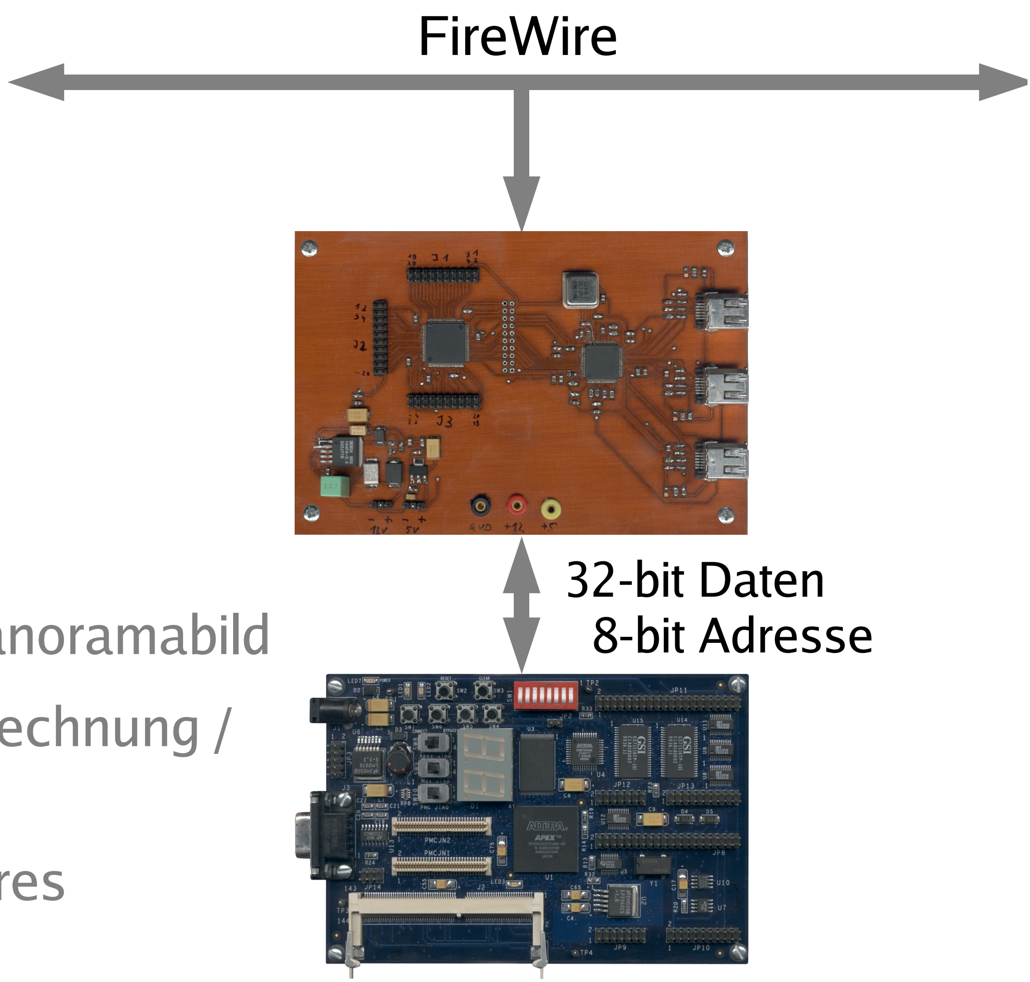
Vorarbeiten und Erfahrungsbericht

Andreas Mäder maeder@informatik.uni-hamburg.de

Inhalt

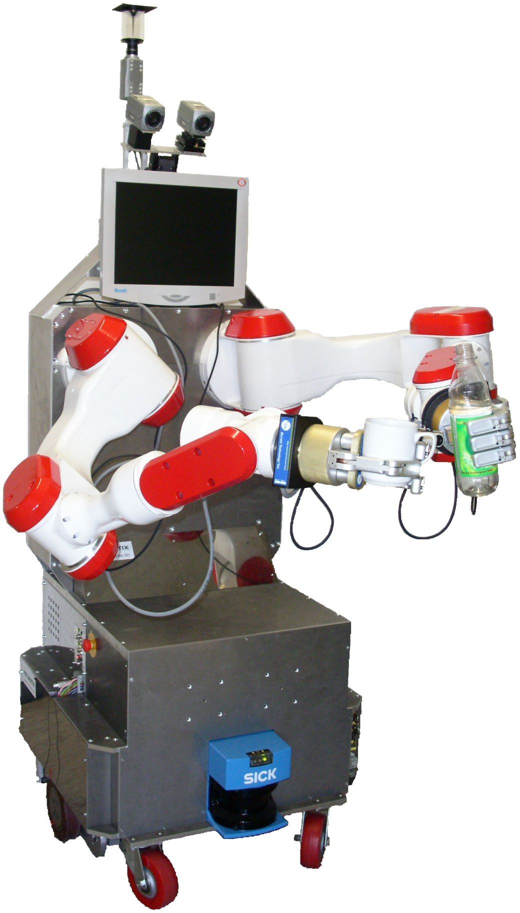
• Motivation	Projektidee
• IEEE1394 – FireWire	Einführung
• Prototypenaufbau	Schnittstellen Hardwareaufbau
• Hardwareentwurf	Systemkomponenten Simulationsmodelle
• Softwareentwurf	Algorithmen Programmierung des Zielsystems
• Probleme	
• Fazit	

Projektidee



FireWire

32-bit Daten
8-bit Adresse



- Autofokus
- Medianfilter
- Omnivision - Panoramabild
- Histogrammberechnung / -Egalisierung
- programmierbares Filter

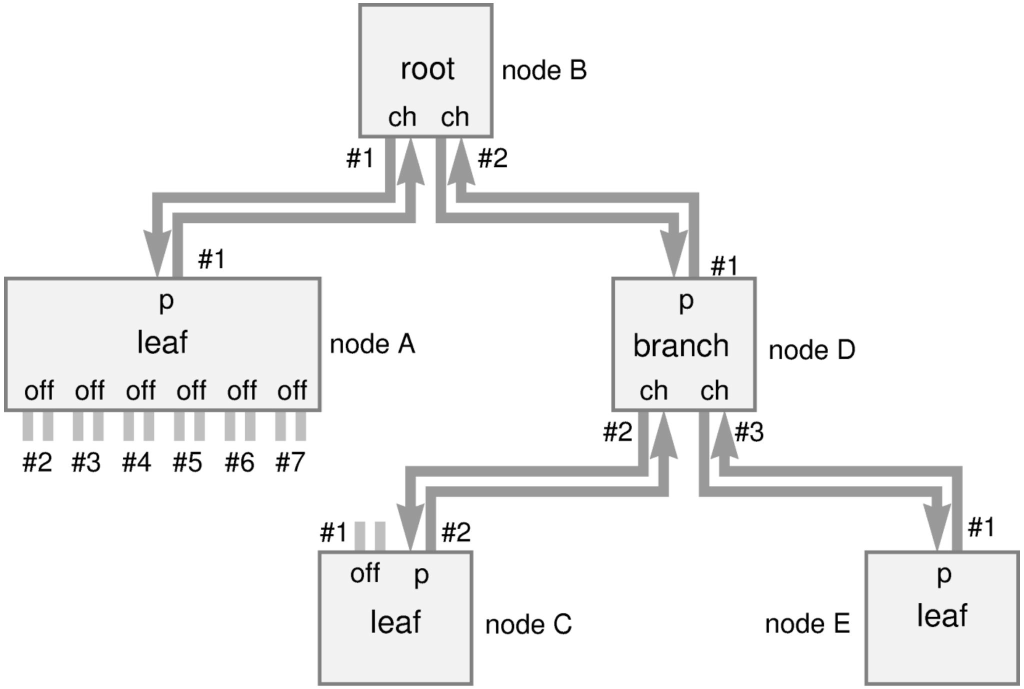
IEEE1394 — FireWire

- Don Anderson: „*FireWire System Architecture: IEEE1394a*“ 2nd Ed., Addison-Wesley, 1999, Reading, Mass., ISBN 0-201-48535-4
- Online-Doku zur Veranstaltung

- Bacc-Arbeiten
  - Sebastian Annies: „*Entwicklung eines FireWire Kamera Interface als eingebettetes System*“
  - Marcin Blaszkowski: „*Entwurf und Implementierung eines FireWire Interface in VHDL*

IEEE1394 – FireWire

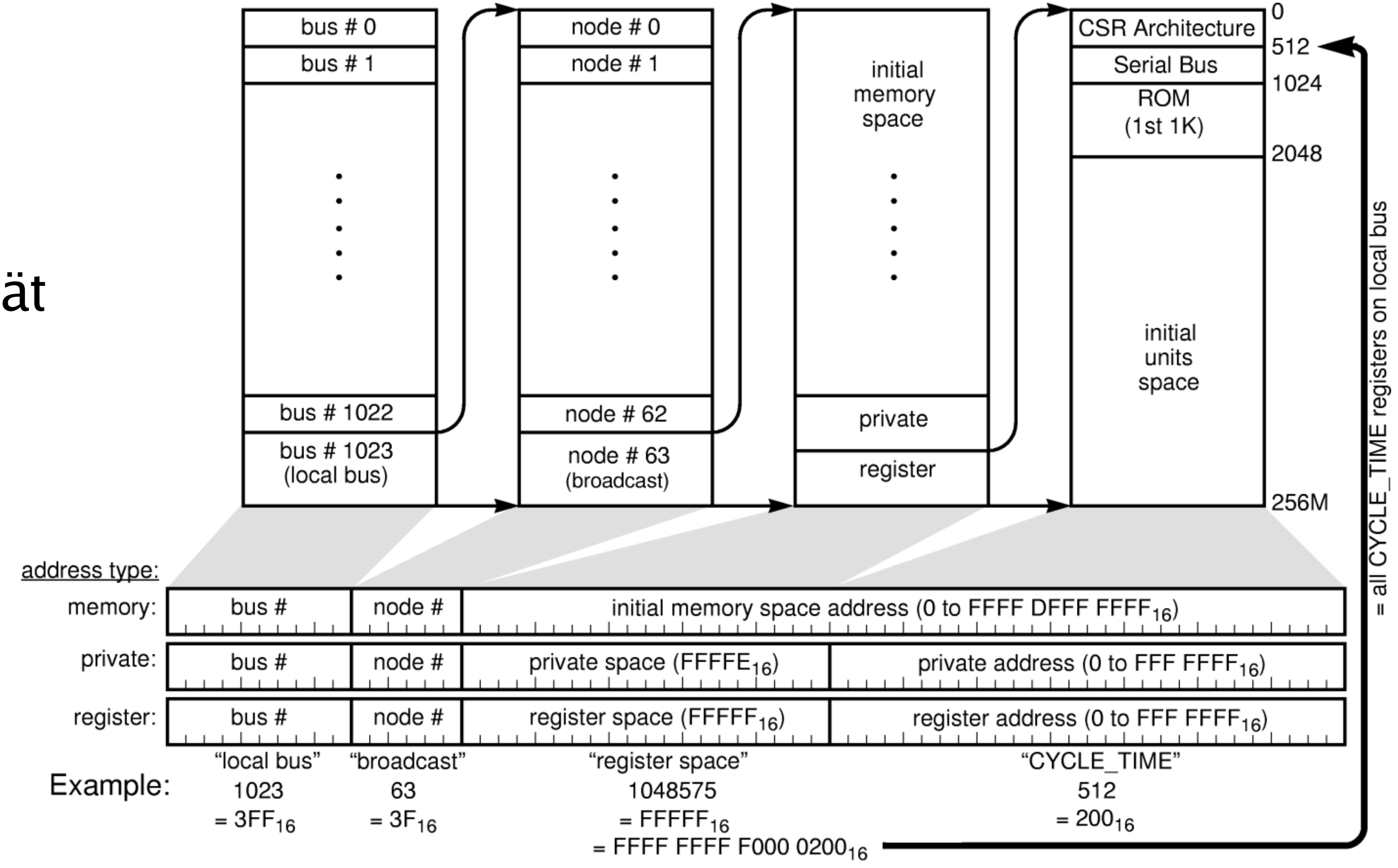
- Standard *„IEEE1394-1995 High Performance Serial Bus“*
  „1394 Based Digital Camera Specification“
  …
- Serielles Bussystem
  - *„hot-pluggable“*
  - mit Spannungsversorgung
  - Geschwindigkeitsklassen
  - topologisch Baum
  - Bus Master
  - Cycle Master


root
node B
ch
ch
#1
#2
#1
p
leaf
node A
off
off
off
off
off
off
#2
#3
#4
#5
#6
#7
#1
p
branch
node D
ch
ch
#2
#3
#1
#2
off
p
leaf
node C
#1
node E
p
leaf

Tree identify complete

IEEE1394 – FireWire

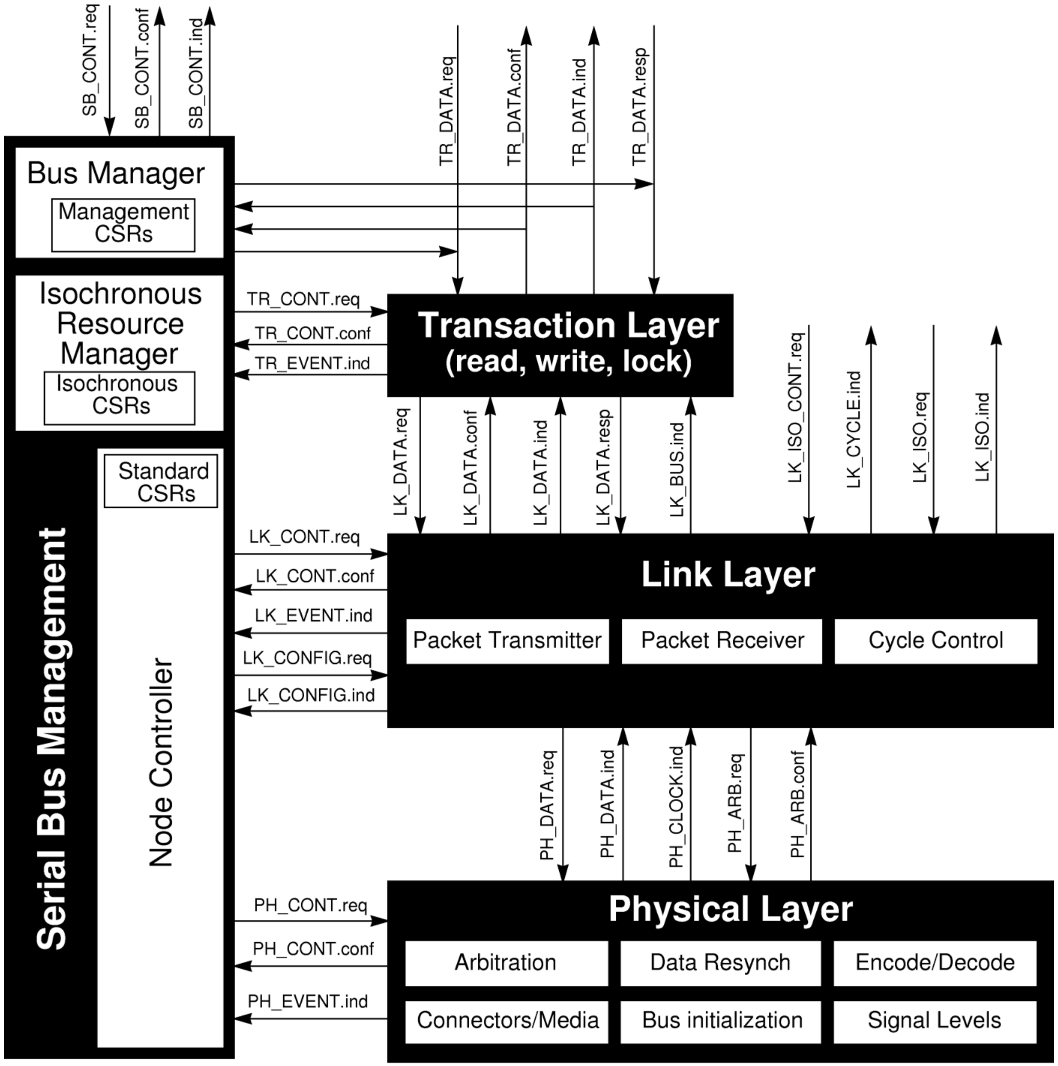
- Addressraum
  - 64 bit
- Knoten ~ Gerät


bus # 0
bus # 1
bus # 1022
bus # 1023
(local bus)
node # 0
node # 1
node # 62
node # 63
(broadcast)
initial
memory
space
private
register
CSR Architecture
Serial Bus
ROM
(1st 1K)
initial
units
space
0
512
1024
2048
256M
= all CYCLE_TIME registers on local bus
address type:
memory:
private:
register:
bus #
node #
initial memory space address (0 to FFFF DFFF FFFF$_{16}$)
private space (FFFFE$_{16}$)
private address (0 to FFF FFFF$_{16}$)
register space (FFFFF$_{16}$)
register address (0 to FFF FFFF$_{16}$)
Example:
“local bus”
1023
= 3FF$_{16}$
“broadcast”
63
= 3F$_{16}$
“register space”
1048575
= FFFFF$_{16}$
“CYCLE_TIME”
512
= 200$_{16}$
= FFFF FFFF F000 0200$_{16}$

Serial Bus addressing

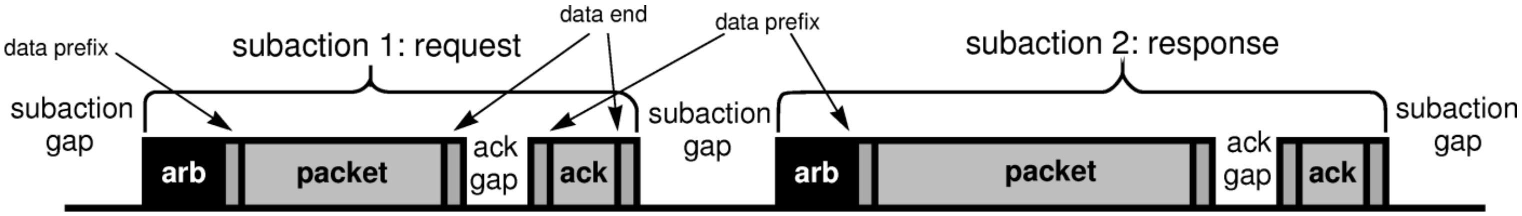
IEEE1394 – FireWire

- Schichtenmodell

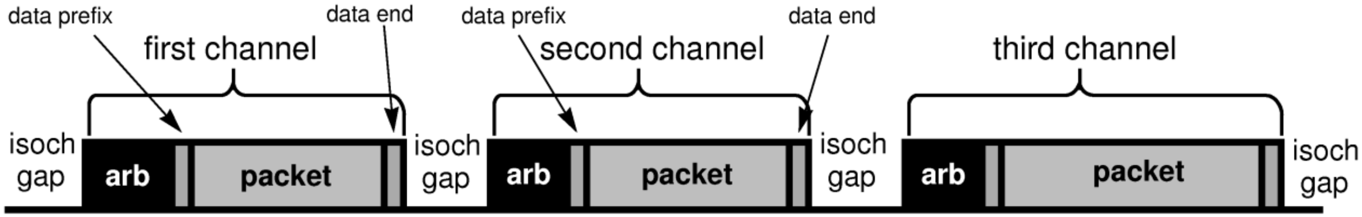

SB_CONT.req
SB_CONT.conf
SB_CONT.ind
TR_DATA.req
TR_DATA.conf
TR_DATA.ind
TR_DATA.resp
Bus Manager
Management CSRs
Isochronous Resource Manager
Isochronous CSRs
Standard CSRs
Serial Bus Management
Node Controller
TR_CONT.req
TR_CONT.conf
TR_EVENT.ind
Transaction Layer (read, write, lock)
LK_DATA.req
LK_DATA.conf
LK_DATA.ind
LK_DATA.resp
LK_BUS.ind
LK_ISO_CONT.req
LK_CYCLE.ind
LK_ISO.req
LK_ISO.ind
LK_CONT.req
LK_CONT.conf
LK_EVENT.ind
LK_CONFIG.req
LK_CONFIG.ind
Link Layer
Packet Transmitter
Packet Receiver
Cycle Control
PH_DATA.req
PH_DATA.ind
PH_CLOCK.ind
PH_ARB.req
PH_ARB.conf
PH_CONT.req
PH_CONT.conf
PH_EVENT.ind
Physical Layer
Arbitration
Data Resynch
Encode/Decode
Connectors/Media
Bus initialization
Signal Levels

IEEE1394 – FireWire

- Transaktionsmodi (Link-Layer)
  - asynchron
  - isochron


data prefix
subaction 1: request
data end
data prefix
subaction 2: response
subaction gap
arb
packet
ack gap
ack
subaction gap
arb
packet
ack gap
ack
subaction gap

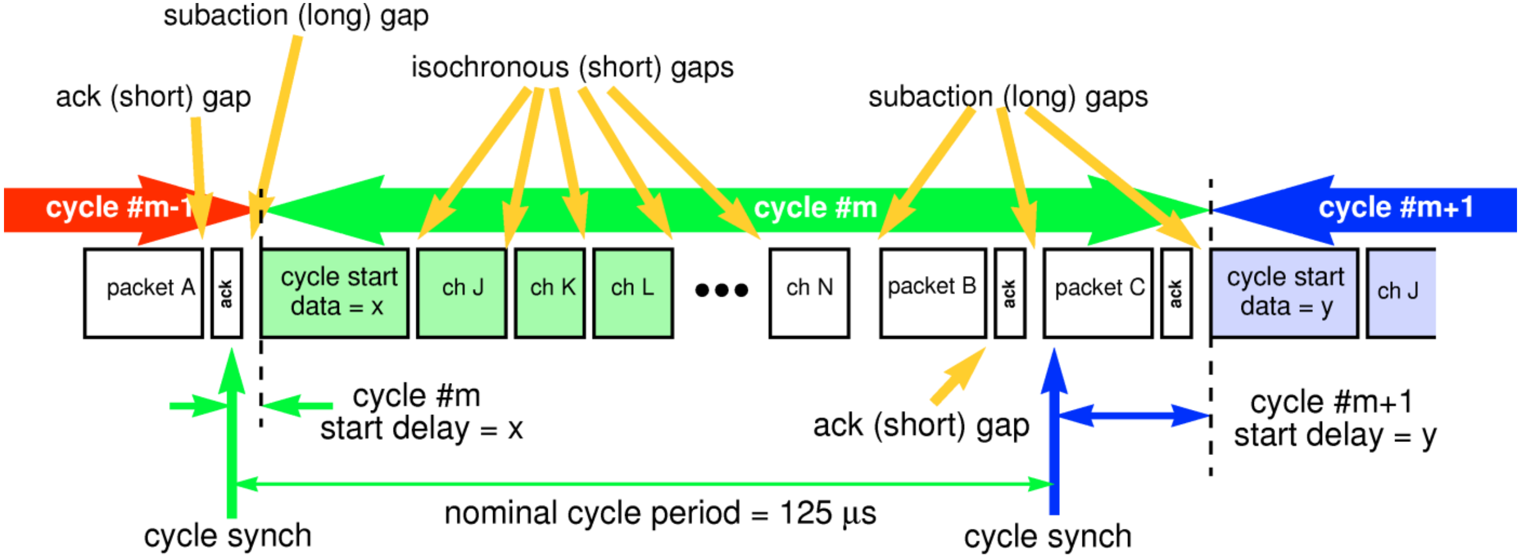
Example asynchronous subactions


data prefix
first channel
data end
data prefix
second channel
data end
third channel
isoch gap
arb
packet
isoch gap
arb
packet
isoch gap
arb
packet
isoch gap

Example isochronous subactions

IEEE1394 – FireWire

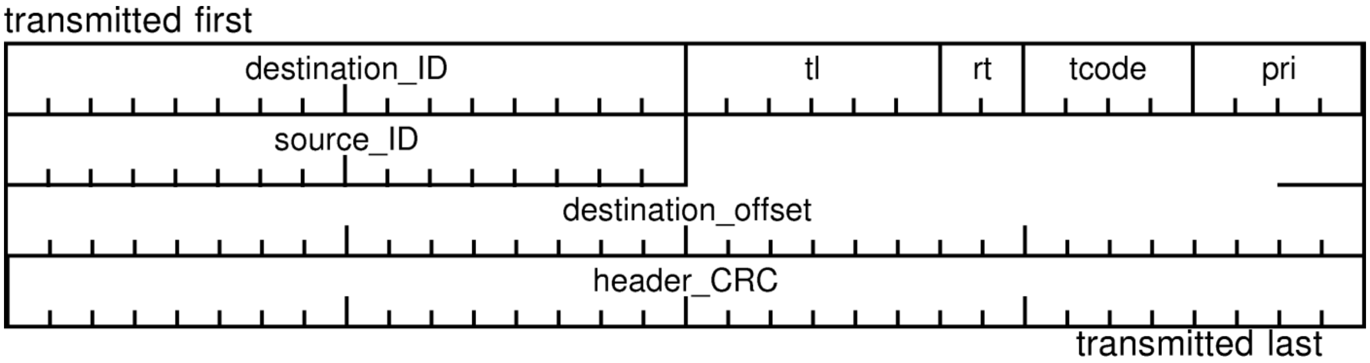
- Zyklen


subaction (long) gap
ack (short) gap
isochronous (short) gaps
subaction (long) gaps
cycle #m-1
cycle #m
cycle #m+1
packet A
ack
cycle start data = x
ch J
ch K
ch L
ch N
packet B
ack
packet C
ack
cycle start data = y
ch J
cycle #m start delay = x
ack (short) gap
cycle #m+1 start delay = y
nominal cycle period = 125 µs
cycle synch
cycle synch

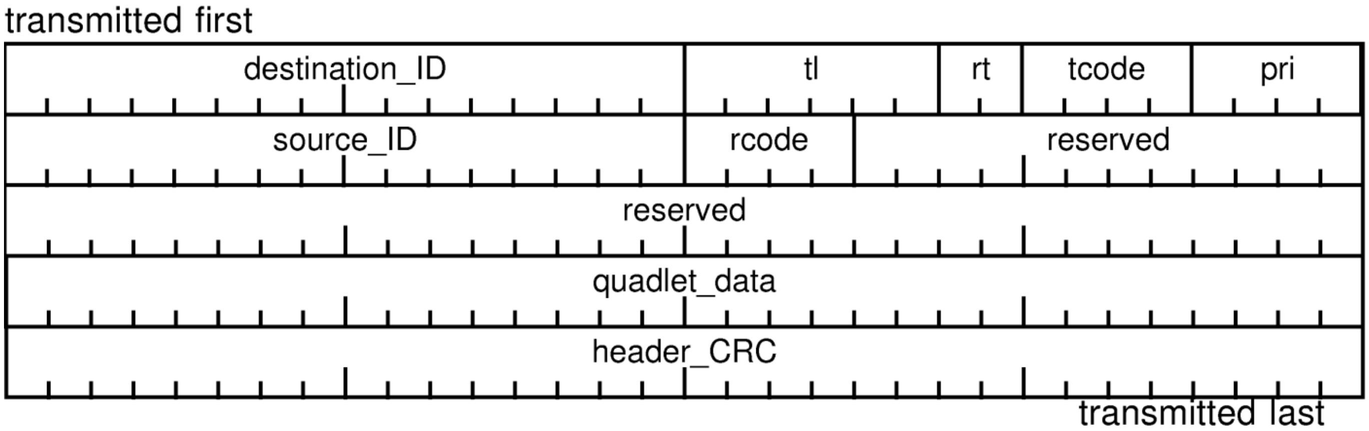
Cycle Structure

IEEE1394 – FireWire

- Formate
  - Quadlet-
  - Paket-orientiert
  - Isochron
  - ...
- Fehlerkorrektur


transmitted first
destination_ID
tl
rt
tcode
pri
source_ID
destination_offset
header_CRC
transmitted last

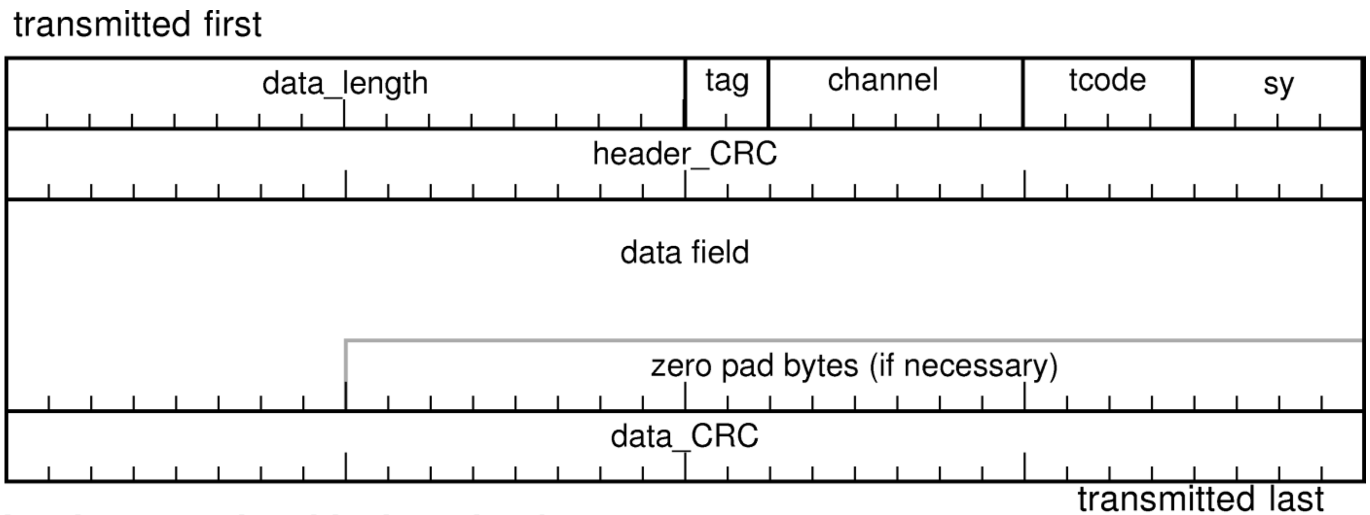
Read request for data quadlet packet format


transmitted first
destination_ID
tl
rt
tcode
pri
source_ID
rcode
reserved
reserved
quadlet_data
header_CRC
transmitted last

Read response for data quadlet packet format

IEEE1394 – FireWire

- Formate


transmitted first
data_length
tag
channel
tcode
sy
header_CRC
data field
zero pad bytes (if necessary)
data_CRC
transmitted last

Isochronous data-block packet format

- TI Link-Layer Chip
  - direkte Übernahme der Formate
  - ohne Fehlerkorrektur

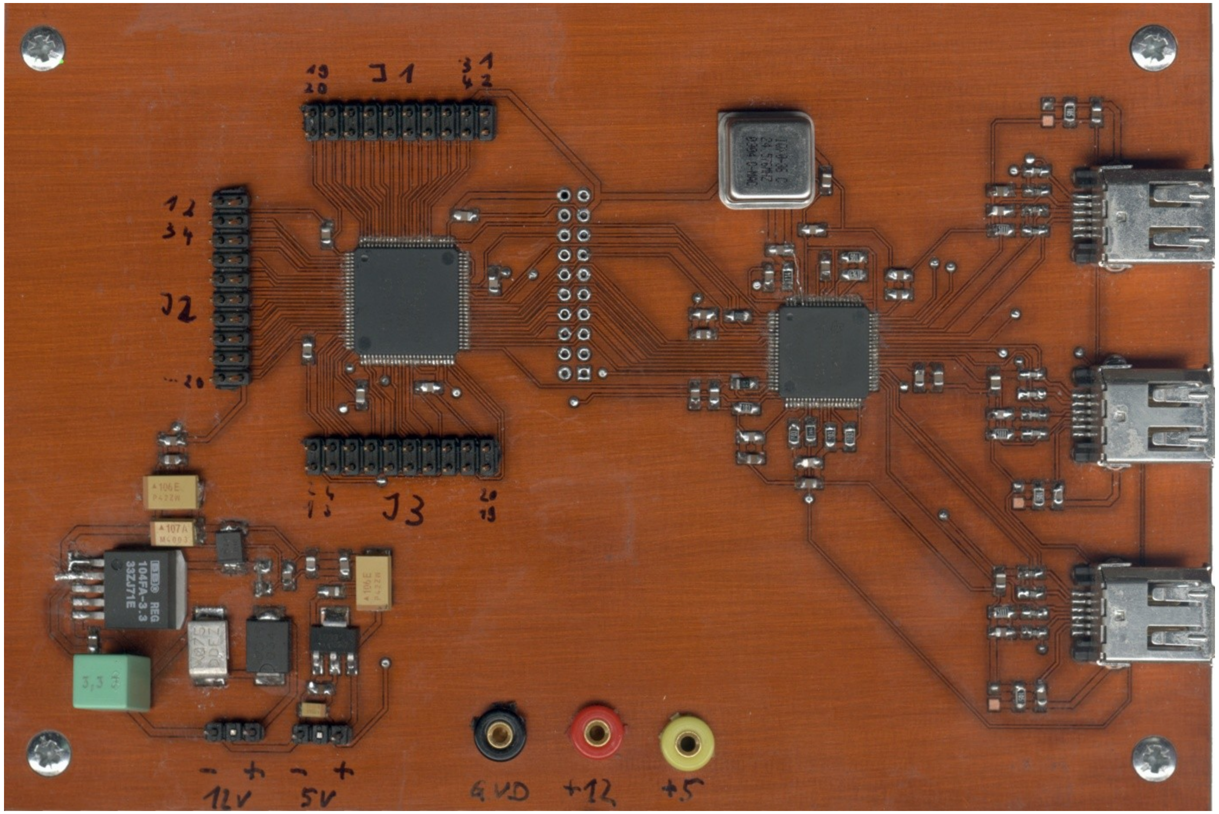
Low-Level Hardware

- FireWire Platine
- TI-Referenzdesign

2-lagig (AB TIS)

TSB12LV01B
link layer controller

TSB41AB3
physical layer chip


19
20
J1
31
42
12
34
J2
20
J3
12V
5V
GND
+12
+5

Low-Level Hardware

- TSB12LV01B
  interner Aufbau

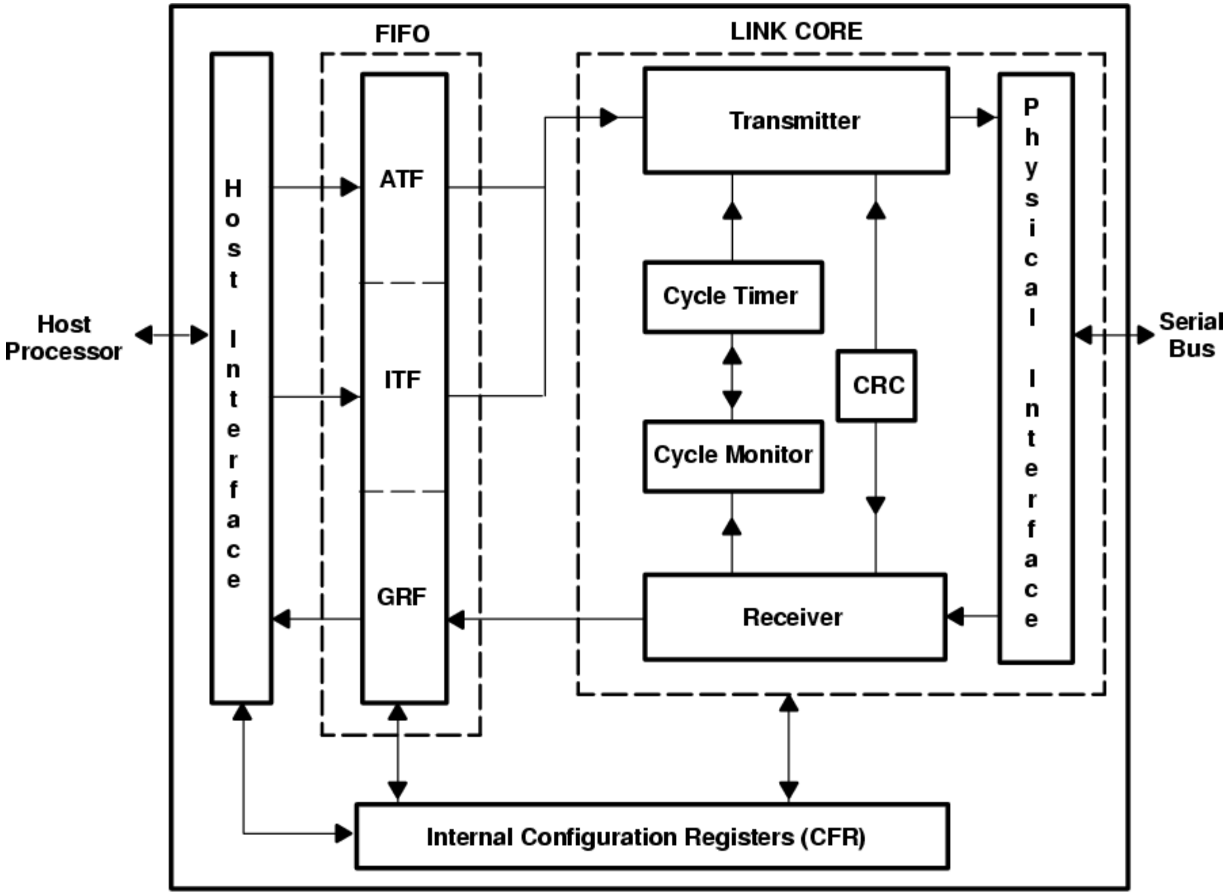

FIFO
LINK CORE
Host Processor
Host Interface
ATF
ITF
GRF
Transmitter
Cycle Timer
Cycle Monitor
CRC
Receiver
Physical Interface
Serial Bus
Internal Configuration Registers (CFR)

Figure 2–1. TSB12LV01B Block Diagram

Low-Level Schnittstelle

- Schnittstellen
  - memory mapped
  - 32-bit Daten
  - 8-bit Adresse

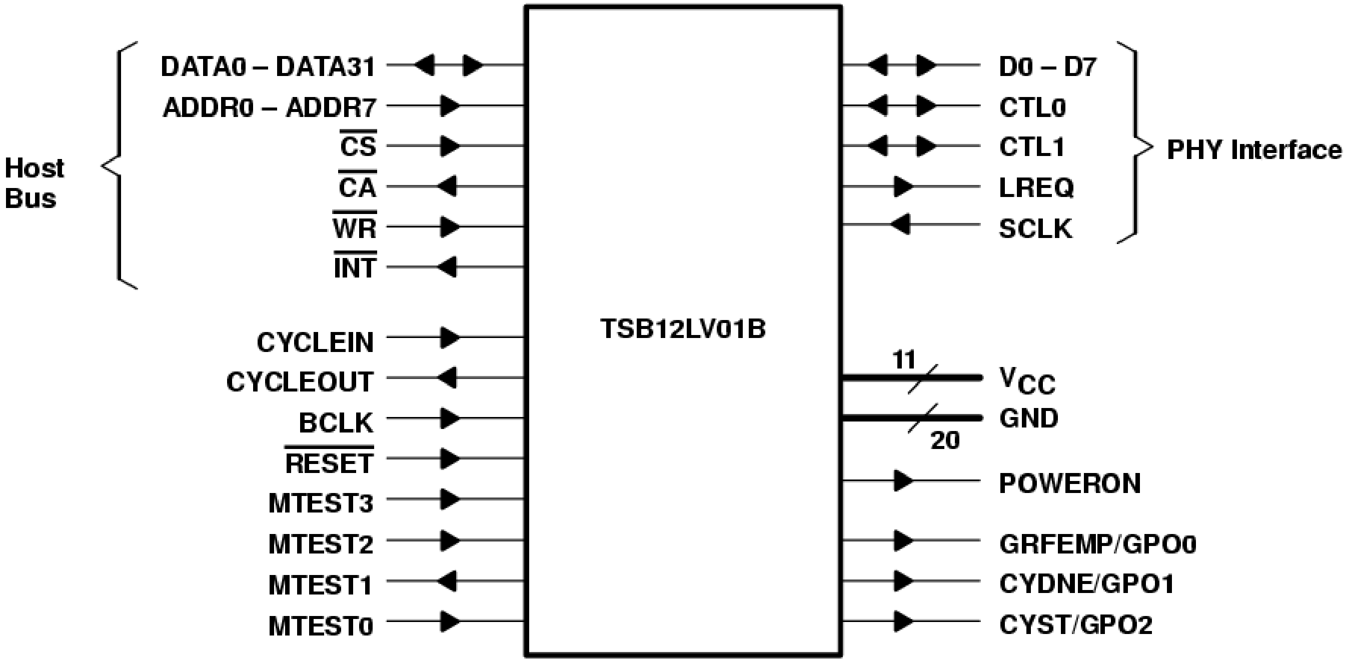
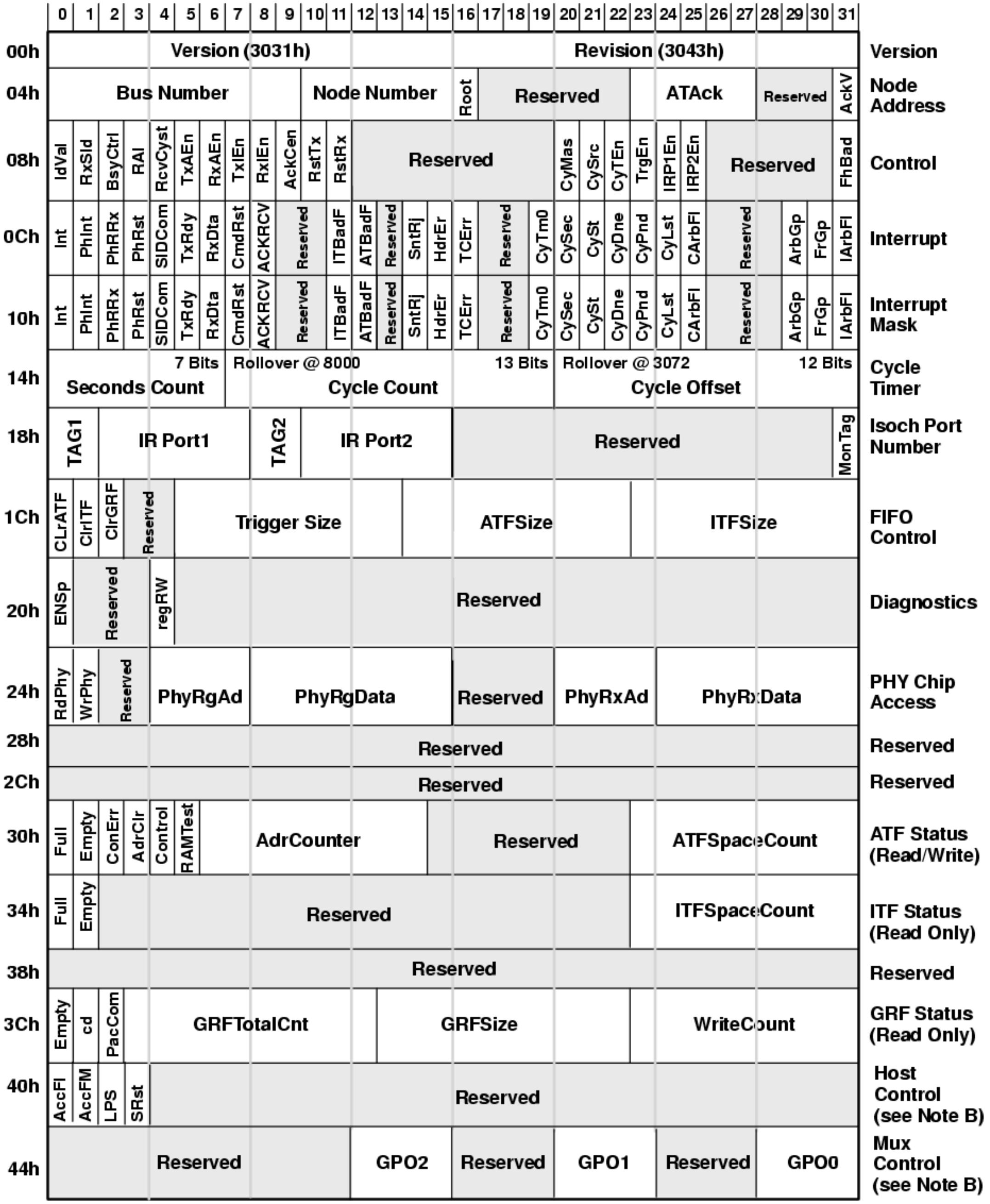

Host Bus
DATA0 – DATA31
ADDR0 – ADDR7
CS
CA
WR
INT
CYCLEIN
CYCLEOUT
BCLK
RESET
MTEST3
MTEST2
MTEST1
MTEST0
TSB12LV01B
D0 – D7
CTL0
CTL1
LREQ
SCLK
PHY Interface
11
VCC
GND
20
POWERON
GRFEMP/GPO0
CYDNE/GPO1
CYST/GPO2

Figure 1–1. TSB12LV01B Terminal Functions

- Speicherlayout
  - Kontrollsektion

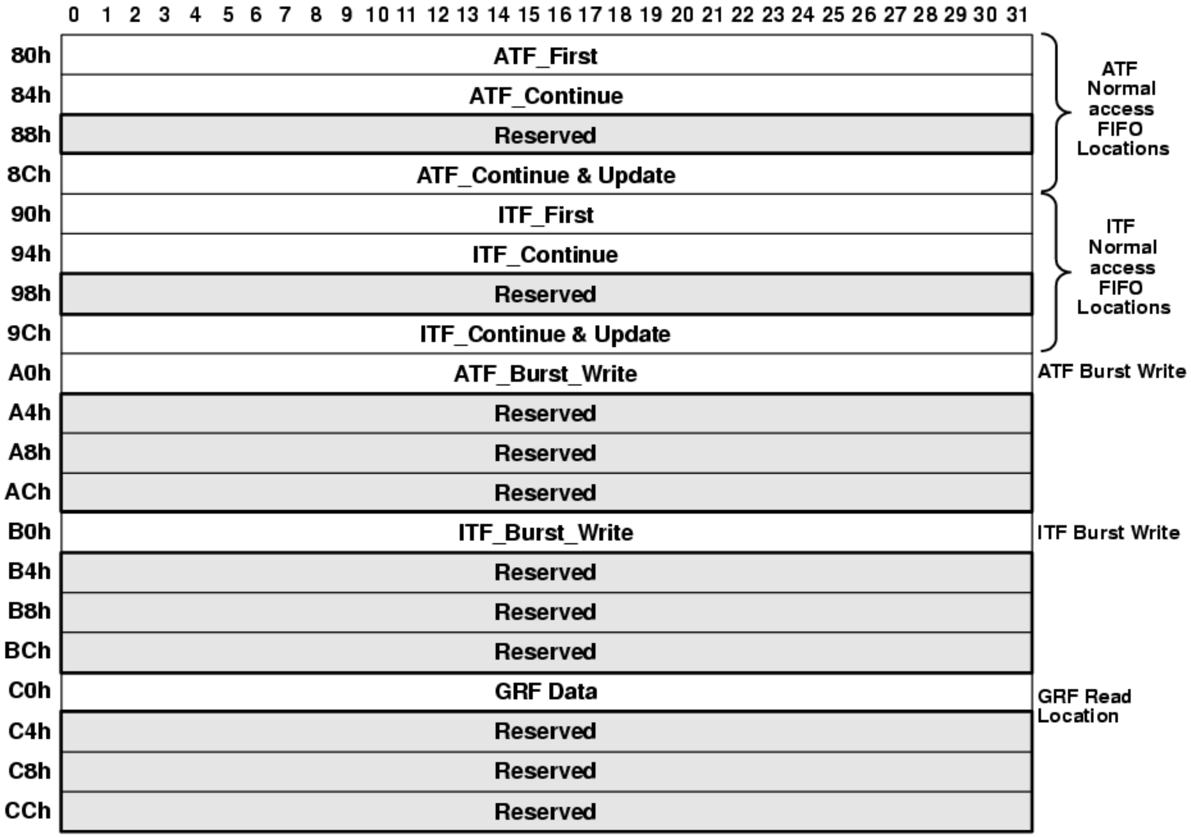


Offset	Bits 0–31	Register
00h	Version (3031h), Revision (3043h)	Version
04h	Bus Number, Node Number, Root, Reserved, ATAck, Reserved, AckV	Node Address
08h	IdVal, RxSld, BsyCtrl, RAI, RcvCyst, TxAEn, RxAEn, TxIEn, RxIEn, AckCen, RstTx, RstRx, Reserved, CyMas, CySrc, CyTEn, TrgEn, IRP1En, IRP2En, Reserved, FhBad	Control
0Ch	Int, PhInt, PhRRx, PhRst, SIDCom, TxRdy, RxDta, CmdRst, ACKRCV, Reserved, ITBadF, ATBadF, Reserved, SntRj, HdrEr, TCErr, Reserved, CyTm0, CySec, CySt, CyDne, CyPnd, CyLst, CArbFl, Reserved, ArbGp, FrGp, IArbFl	Interrupt
10h	Int, PhInt, PhRRx, PhRst, SIDCom, TxRdy, RxDta, CmdRst, ACKRCV, Reserved, ITBadF, ATBadF, Reserved, SntRj, HdrEr, TCErr, Reserved, CyTm0, CySec, CySt, CyDne, CyPnd, CyLst, CArbFl, Reserved, ArbGp, FrGp, IArbFl	Interrupt Mask
14h	Seconds Count (7 Bits), Cycle Count (Rollover @ 8000, 13 Bits), Cycle Offset (Rollover @ 3072, 12 Bits)	Cycle Timer
18h	TAG1, IR Port1, TAG2, IR Port2, Reserved, MonTag	Isoch Port Number
1Ch	CLrATF, ClrITF, ClrGRF, Reserved, Trigger Size, ATFSize, ITFSize	FIFO Control
20h	ENSp, Reserved, regRW, Reserved	Diagnostics
24h	RdPhy, WrPhy, Reserved, PhyRgAd, PhyRgData, Reserved, PhyRxAd, PhyRxData	PHY Chip Access
28h	Reserved	Reserved
2Ch	Reserved	Reserved
30h	Full, Empty, ConErr, AdrClr, Control, RAMTest, AdrCounter, Reserved, ATFSpaceCount	ATF Status (Read/Write)
34h	Full, Empty, Reserved, ITFSpaceCount	ITF Status (Read Only)
38h	Reserved	Reserved
3Ch	Empty, cd, PacCom, GRFTotalCnt, GRFSize, WriteCount	GRF Status (Read Only)
40h	AccFl, AccFM, LPS, SRst, Reserved	Host Control (see Note B)
44h	Reserved, GPO2, Reserved, GPO1, Reserved, GPO0	Mux Control (see Note B)

NOTES: A. All gray areas (bits) are reserved bits.
B. This register is new to the TSB12LV01B and does not exist in the TSB12LV01A.

Figure 3–1. Internal Register Map

- Speicherlayout
  - FIFOs

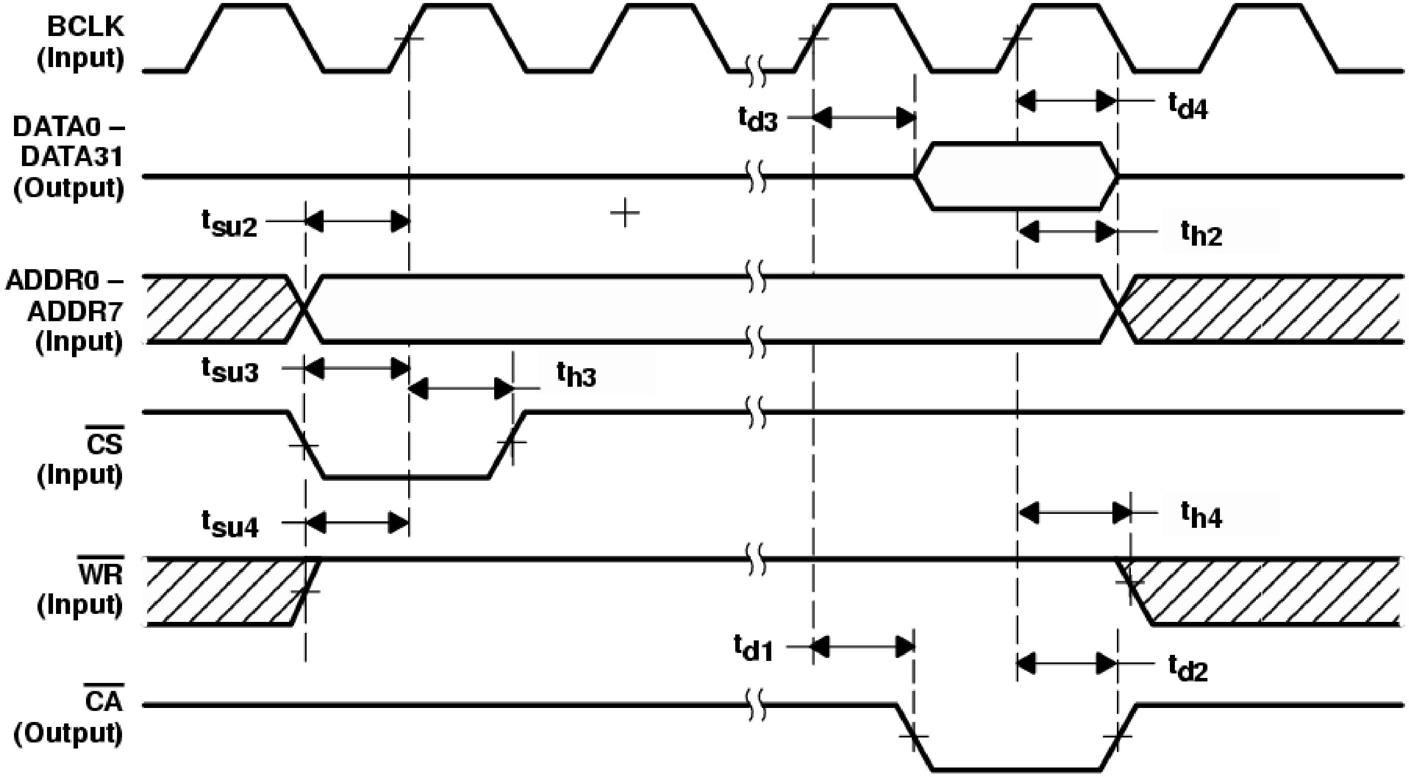


	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	
80h	**ATF_First**	ATF Normal access FIFO Locations
84h	**ATF_Continue**	
88h	**Reserved**	
8Ch	**ATF_Continue & Update**	
90h	**ITF_First**	ITF Normal access FIFO Locations
94h	**ITF_Continue**	
98h	**Reserved**	
9Ch	**ITF_Continue & Update**	
A0h	**ATF_Burst_Write**	ATF Burst Write
A4h	**Reserved**	
A8h	**Reserved**	
ACh	**Reserved**	
B0h	**ITF_Burst_Write**	ITF Burst Write
B4h	**Reserved**	
B8h	**Reserved**	
BCh	**Reserved**	
C0h	**GRF Data**	GRF Read Location
C4h	**Reserved**	
C8h	**Reserved**	
CCh	**Reserved**	

Figure 3–3. TSB12LV01B Controller-FIFO-Access Address Map

Low-Level Schnittstelle

- Read
  0x00 - 0x2C

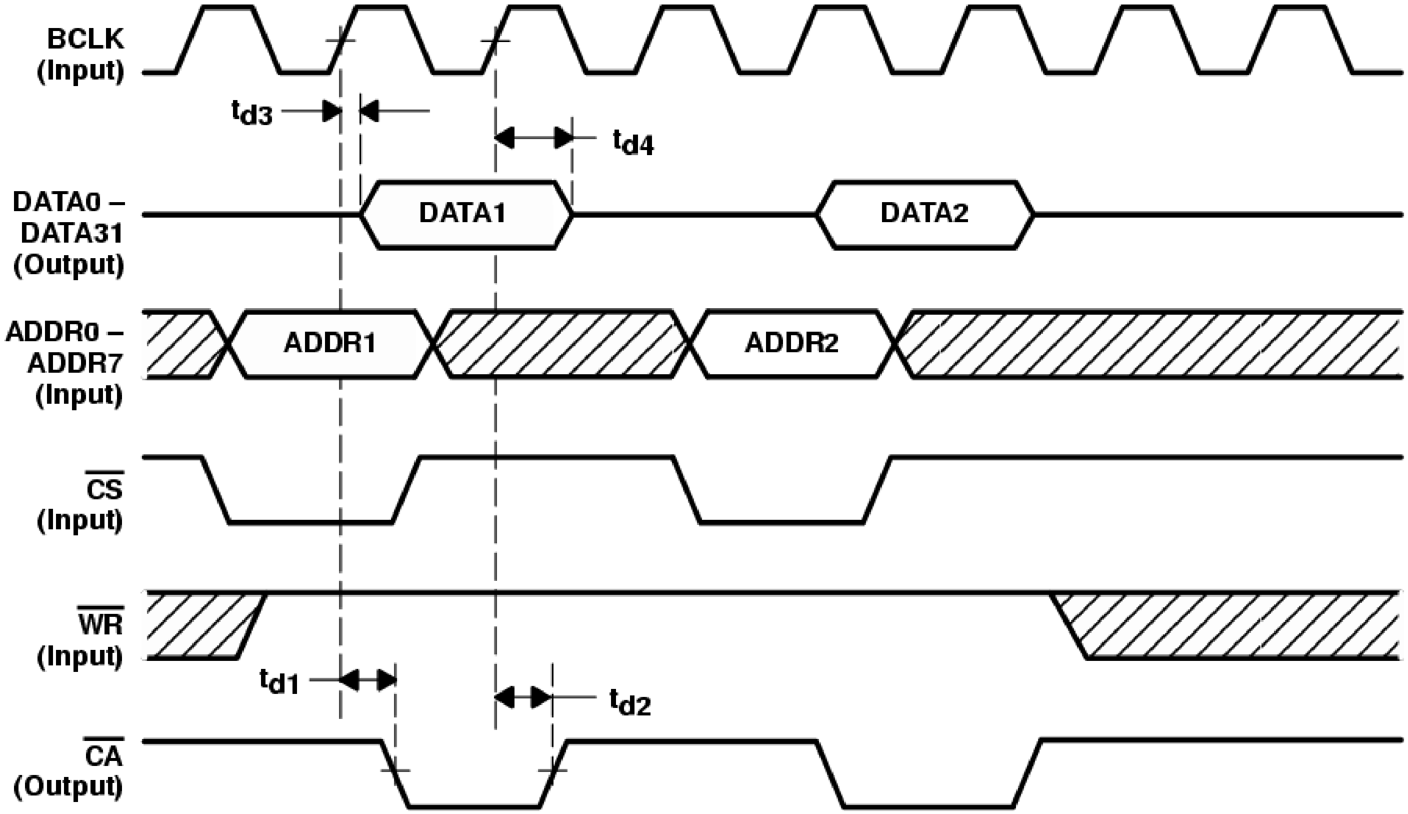

BCLK (Input)
DATA0 – DATA31 (Output)
ADDR0 – ADDR7 (Input)
CS (Input)
WR (Input)
CA (Output)
td3
td4
tsu2
th2
tsu3
th3
tsu4
th4
td1
td2

NOTE A. Following a $\overline{CS}$ assertion, there may be a maximum of 9 rising edges of BCLK before a $\overline{CA}$ is returned. $\overline{CA}$ must be returned before another $\overline{CS}$ may be asserted.

Figure 6–3. Host-Interface Read-Cycle Waveforms (Address: 00h – 2Ch)

Low-Level Schnittstelle

- Read
  0x30 - ...

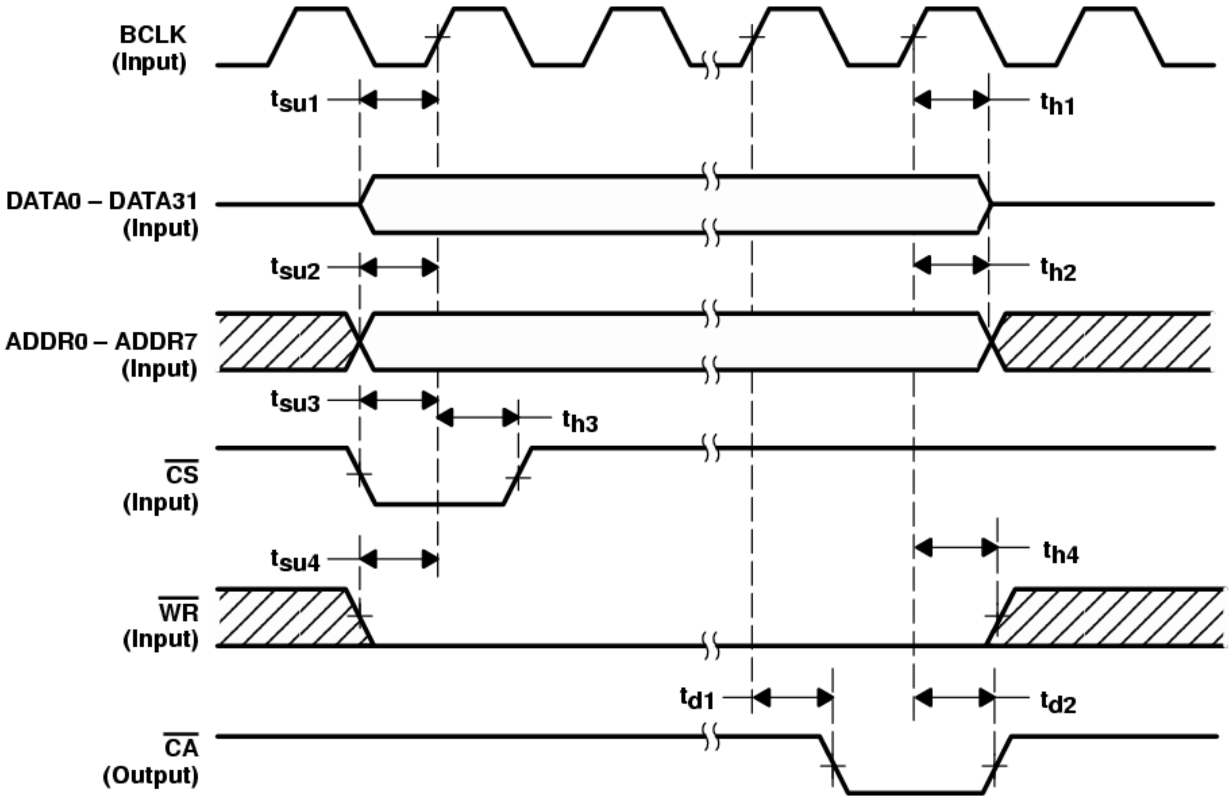

BCLK
(Input)
td3
td4
DATA0 –
DATA31
(Output)
DATA1
DATA2
ADDR0 –
ADDR7
(Input)
ADDR1
ADDR2
$\overline{CS}$
(Input)
$\overline{WR}$
(Input)
td1
td2
$\overline{CA}$
(Output)

NOTE A. There must be a minimum of 3 rising edges of BCLK between assertions of $\overline{CS}$.

Figure 6–5. Host-Interface Quick Read-Cycle Waveforms (ADDRESS ≥ 30h)

Low-Level Schnittstelle

- Write
  0x00 - 0x2C

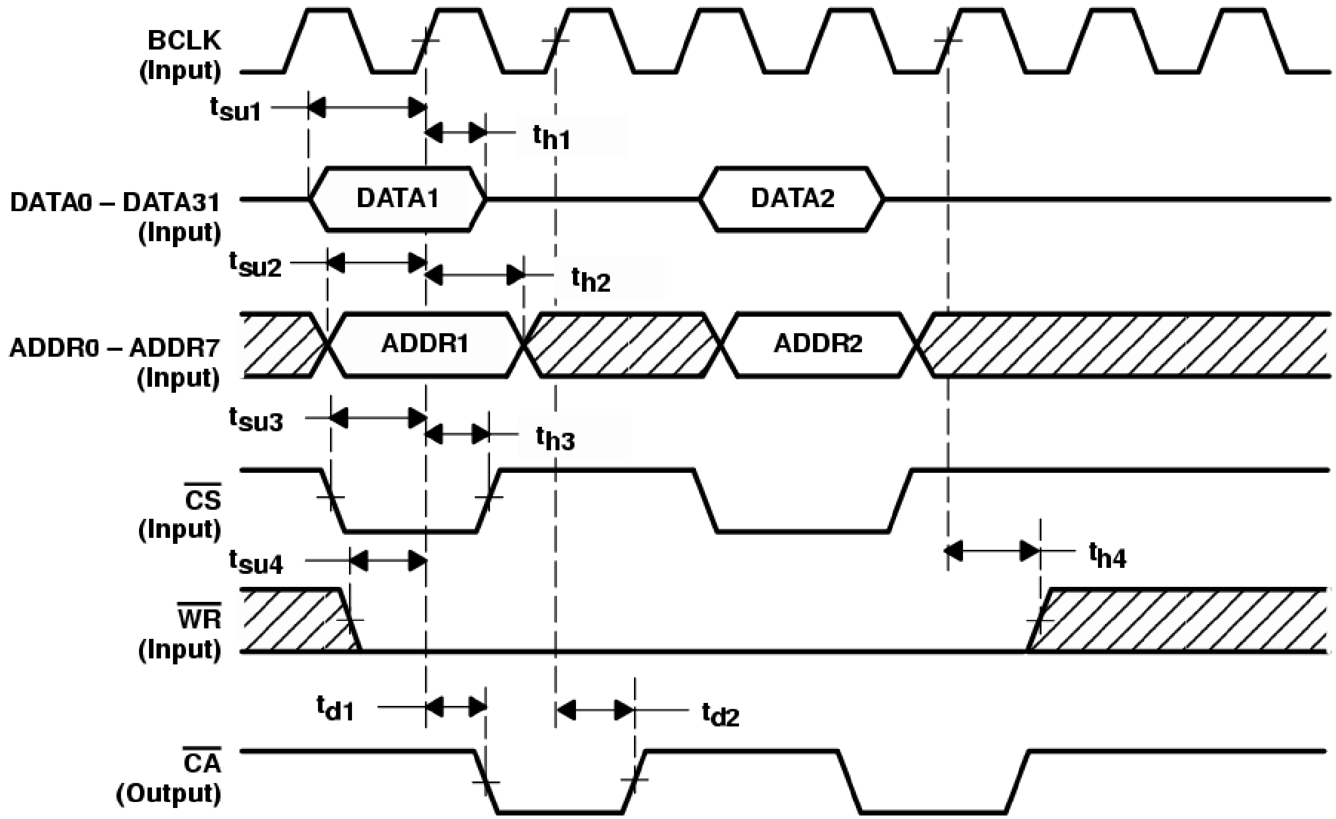

BCLK (Input)
tsu1
th1
DATA0 – DATA31 (Input)
tsu2
th2
ADDR0 – ADDR7 (Input)
tsu3
th3
CS (Input)
tsu4
th4
WR (Input)
td1
td2
CA (Output)

NOTE A. Following a $\overline{\text{CS}}$ assertion, there may be a maximum of 9 rising edges of BCLK before a $\overline{\text{CA}}$ is returned. $\overline{\text{CA}}$ must be returned before another $\overline{\text{CS}}$ may be asserted.

Figure 6–2. Host-Interface Write-Cycle Waveforms (Address: 00h – 2Ch)

Low-Level Schnittstelle

- Write
  0x30 - ...

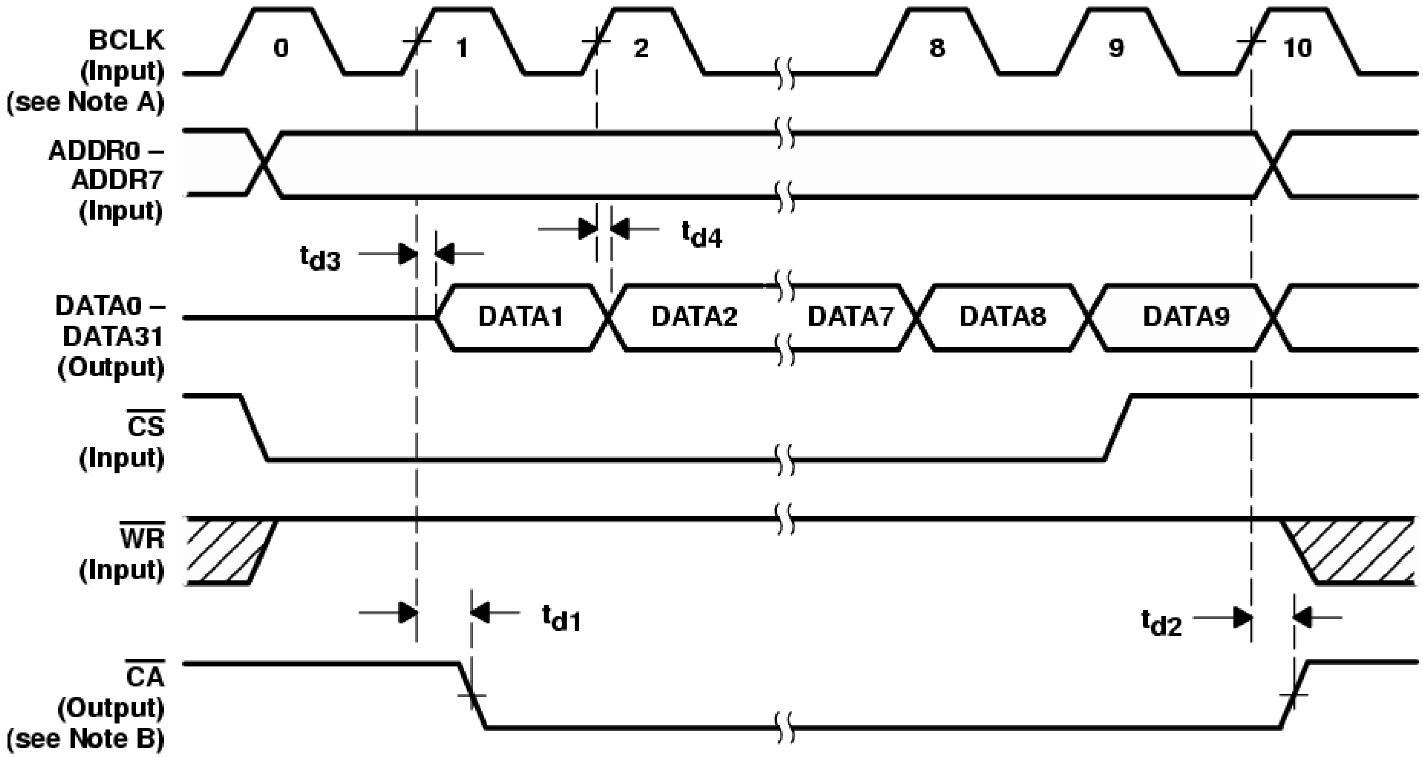

BCLK (Input)
tsu1
th1
DATA0 – DATA31 (Input)
DATA1
DATA2
tsu2
th2
ADDR0 – ADDR7 (Input)
ADDR1
ADDR2
tsu3
th3
CS (Input)
tsu4
th4
WR (Input)
td1
td2
CA (Output)

NOTE A. There must be a minimum of 3 rising edges of BCLK between assertions of $\overline{\text{CS}}$.

Figure 6–4. Host-Interface Quick Write-Cycle Waveforms (Address ≥ 30h)

Low-Level Schnittstelle

- Read - burst
  0xC0

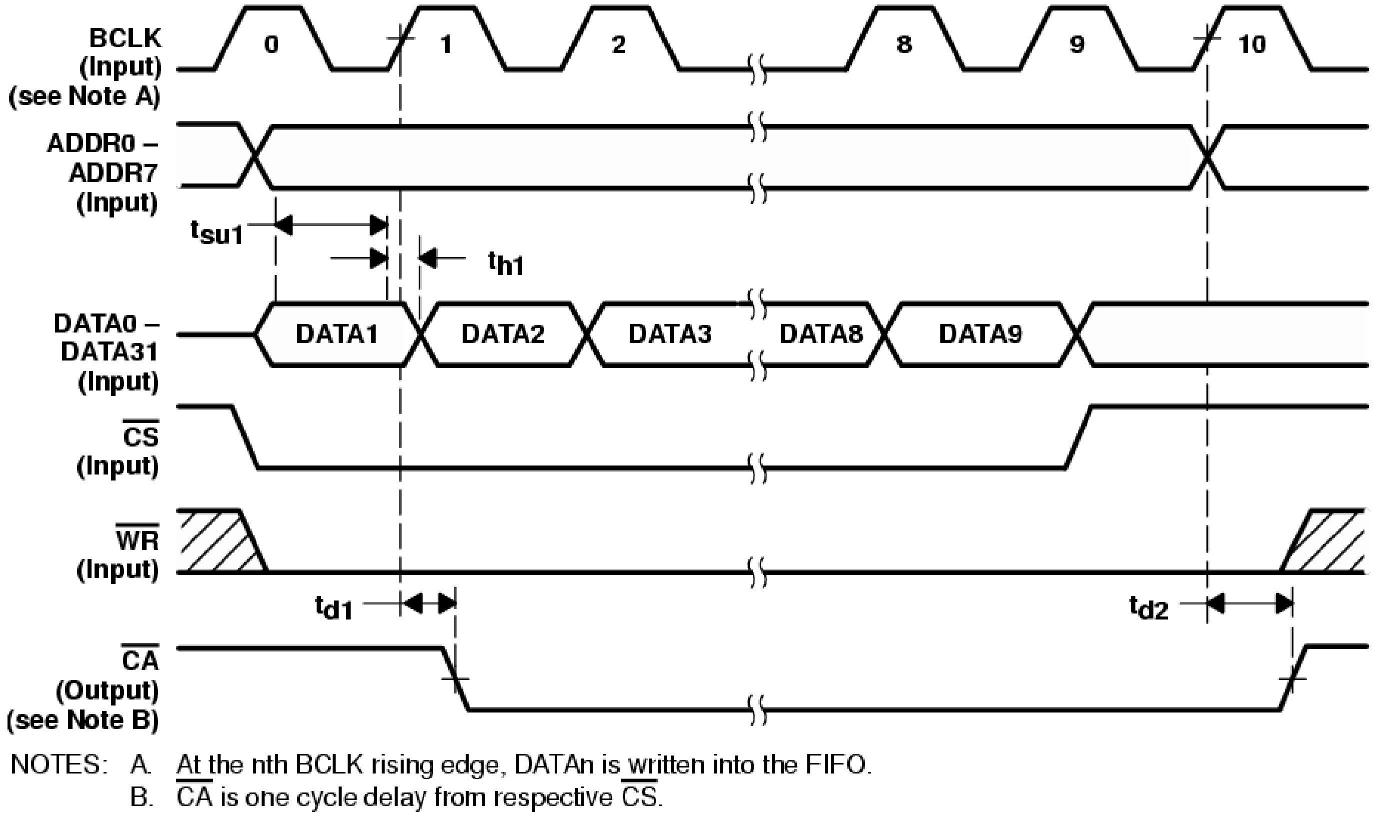

BCLK
(Input)
(see Note A)
0
1
2
8
9
10
ADDR0 –
ADDR7
(Input)
t_{d3}
t_{d4}
DATA0 –
DATA31
(Output)
DATA1
DATA2
DATA7
DATA8
DATA9
$\overline{CS}$
(Input)
$\overline{WR}$
(Input)
t_{d1}
t_{d2}
$\overline{CA}$
(Output)
(see Note B)

NOTES:
A. At the (nth+1) BCLK rising edge, host bus should latch DATAn.
B. $\overline{CA}$ is one cycle delay from respective $\overline{CS}$.
C. These waveforms only apply to address C0h.

Figure 6–7. Burst Read Waveforms

Low-Level Schnittstelle

- Write - burst
  0xA0
  0xB0


BCLK
(Input)
(see Note A)
0
1
2
8
9
10
ADDR0 –
ADDR7
(Input)
t_{su1}
t_{h1}
DATA0 –
DATA31
(Input)
DATA1
DATA2
DATA3
DATA8
DATA9
$\overline{CS}$
(Input)
$\overline{WR}$
(Input)
t_{d1}
t_{d2}
$\overline{CA}$
(Output)
(see Note B)

NOTES: A. At the nth BCLK rising edge, DATAn is written into the FIFO.
B. $\overline{CA}$ is one cycle delay from respective $\overline{CS}$.

Figure 6–6. Burst Write Waveforms

`ioFsm`

- Anforderungen
  - Reaktion innerhalb eines Taktes

    ⇒ Kopplung der Automaten durch *Vorder- und Rückflanke*
  - Bussystem: `data`

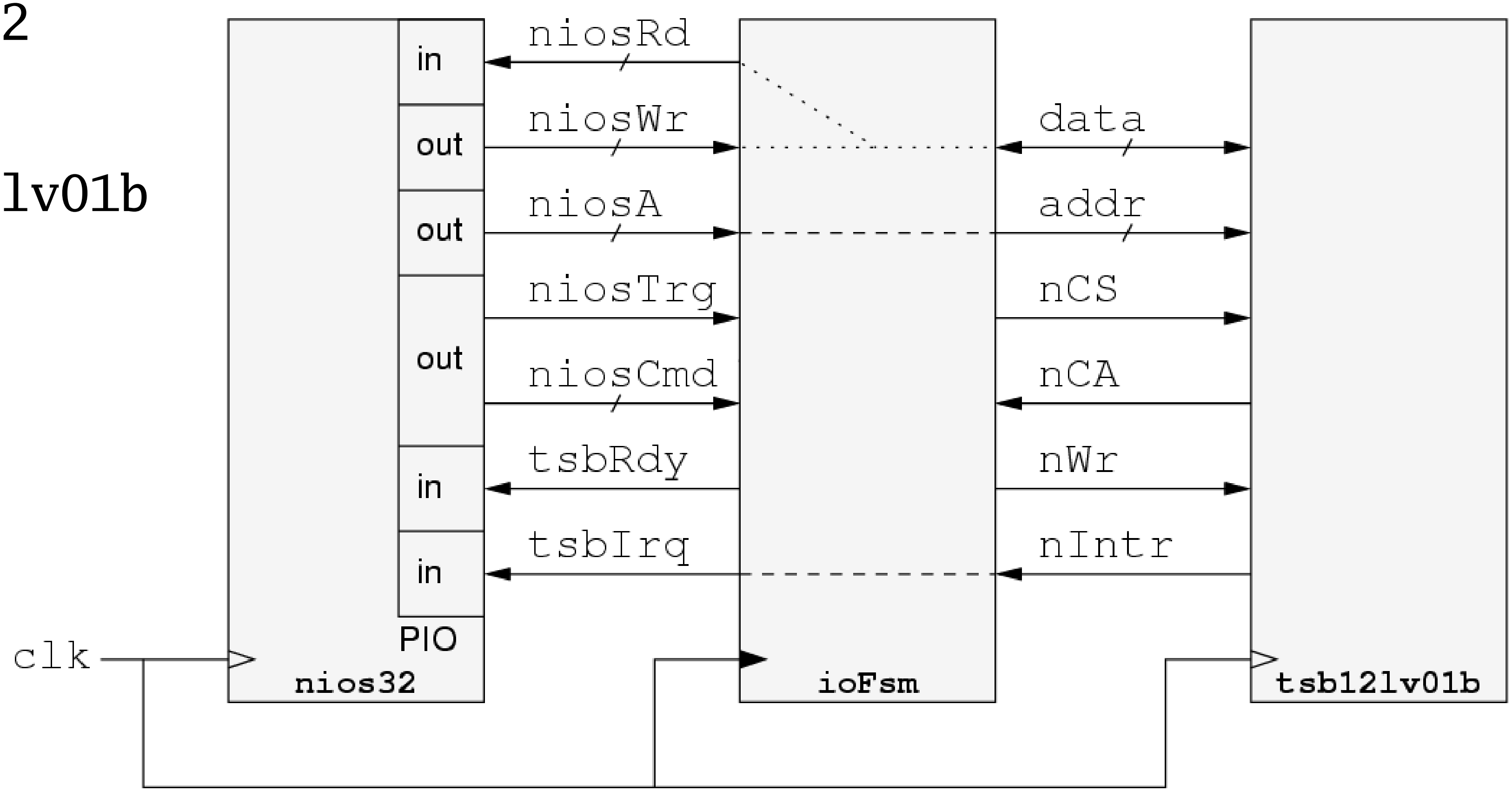
    ⇒ Tristate-Treiber

⇒ Hardware Interface

- Auswertung der Daten
  - Prototyp: Software
  - Applikation: Hardware (400Mbit/s ≈ 2,6Takte, 32bit@33MHz)

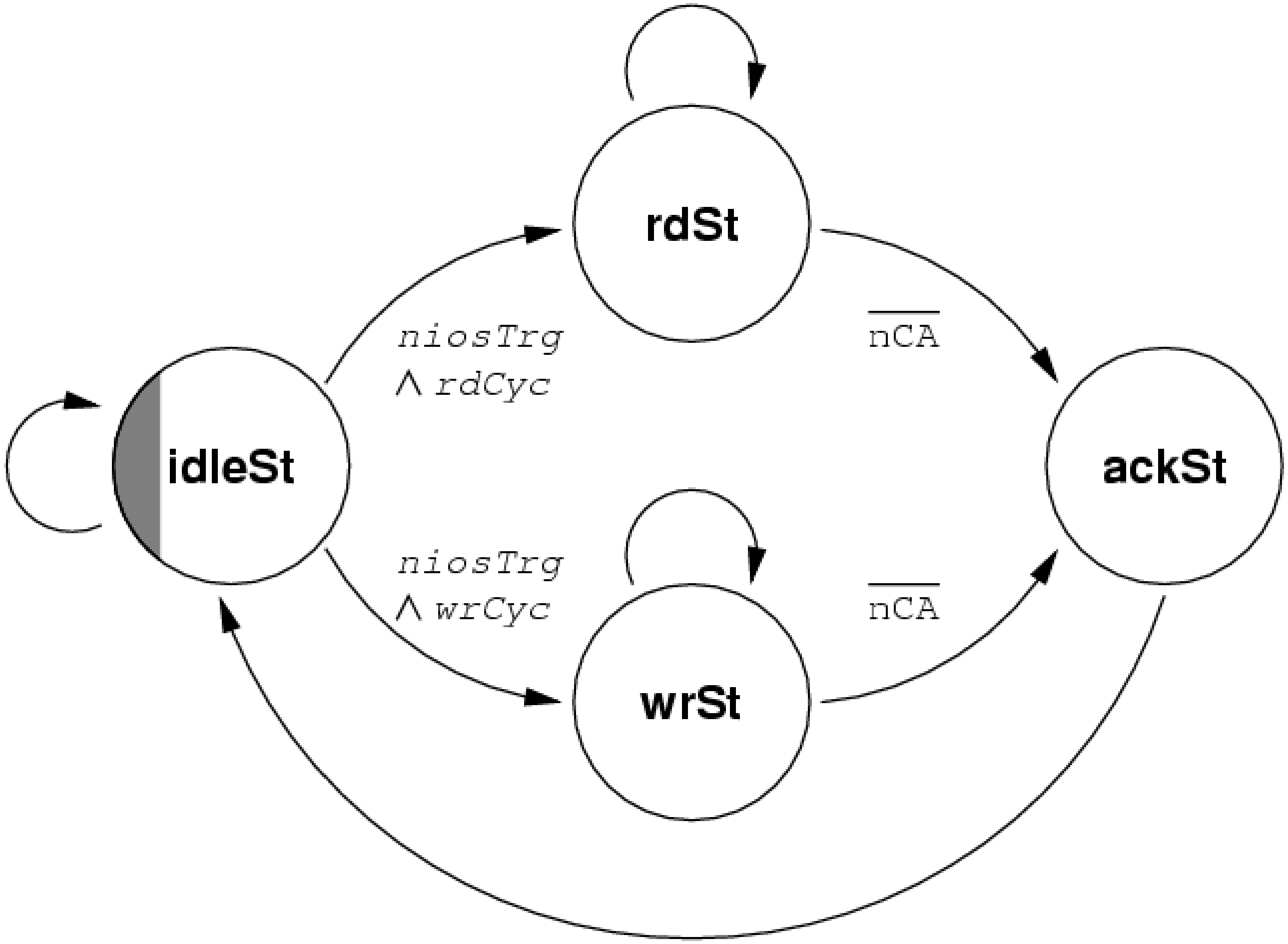
`ioFsm` - Schnittstelle

- Prototyp
  - `nios32`
  - `ioFsm`
  - `tsb12lv01b`

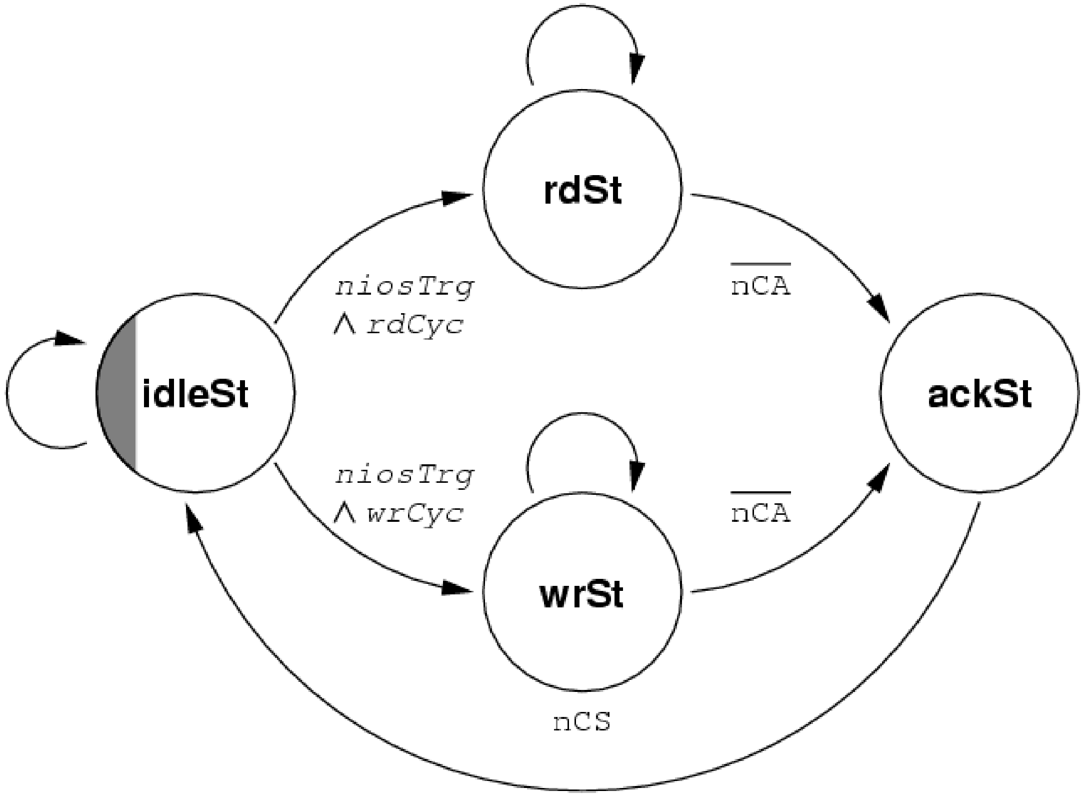

in
out
out
out
in
in
PIO
nios32
niosRd
niosWr
niosA
niosTrg
niosCmd
tsbRdy
tsbIrq
ioFsm
data
addr
nCS
nCA
nWr
nIntr
tsb12lv01b
clk

`ioFsm` - Funktion

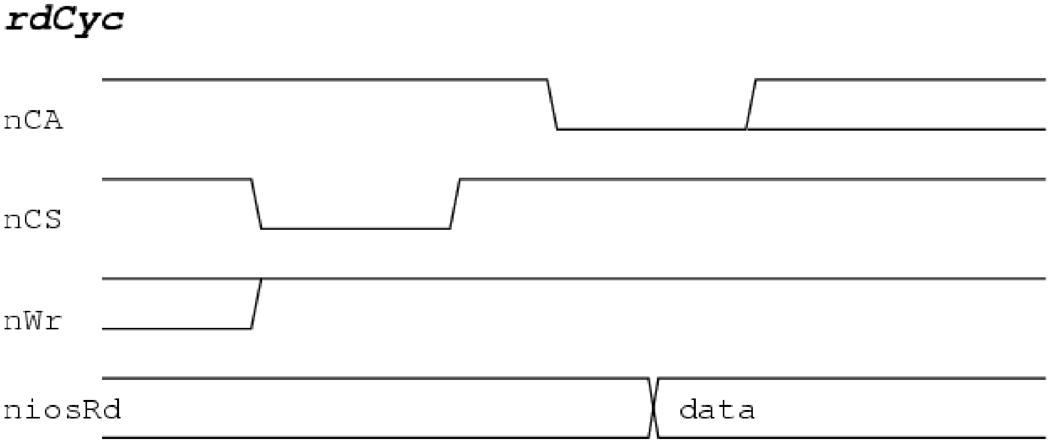
- `ioFsm`
  - Rückflanke
  - Kontrollsignale
    `nCS`, `nWr`, `nCA`
  - Tristate-Treiber
  - Bus aufsplitten


rdSt
idleSt
ackSt
wrSt
niosTrg ∧ rdCyc
niosTrg ∧ wrCyc
$\overline{\text{nCA}}$
$\overline{\text{nCA}}$

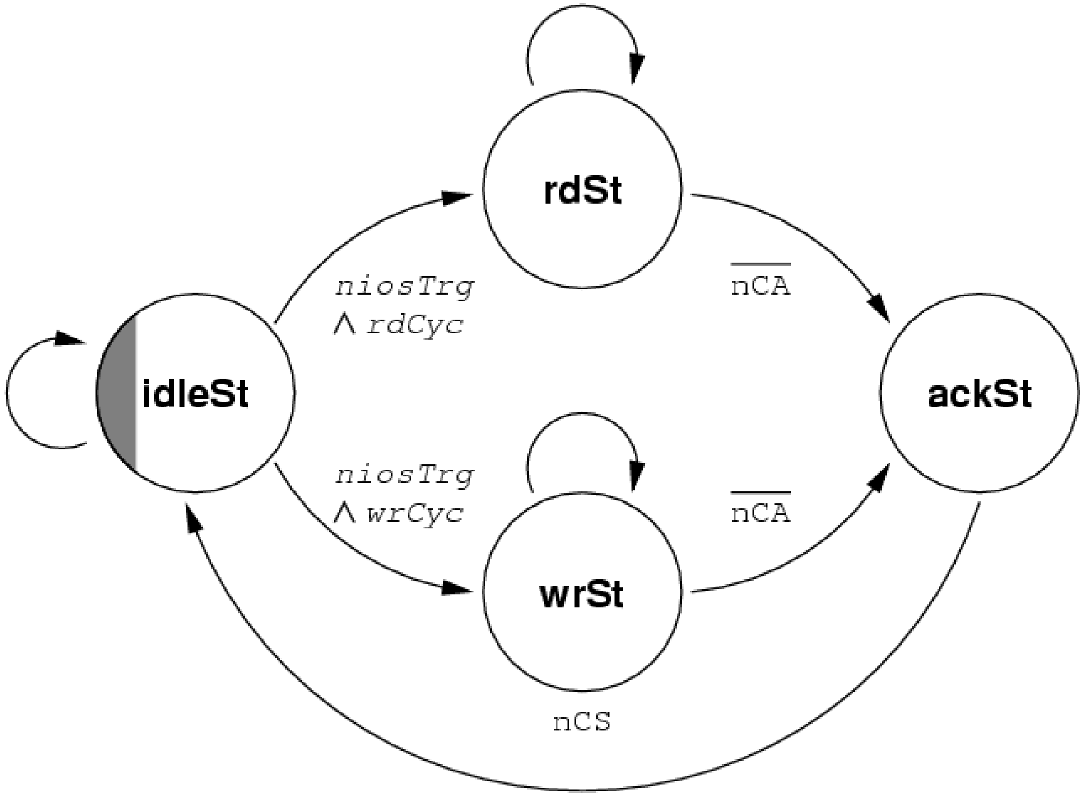
ioFsm


rdSt
idleSt
ackSt
wrSt
niosTrg
∧ rdCyc
niosTrg
∧ wrCyc
$\overline{\text{nCA}}$
$\overline{\text{nCA}}$
nCS

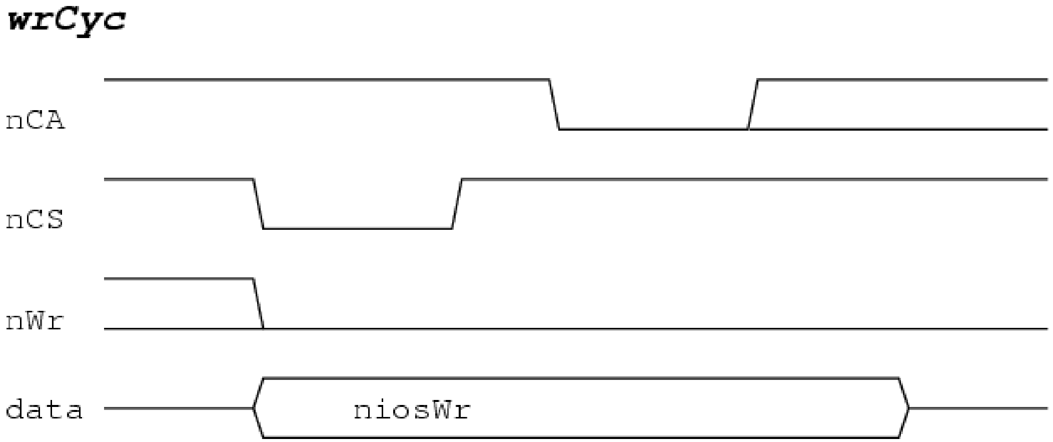
- Lesezyklus


rdCyc
nCA
nCS
nWr
niosRd
data

ioFsm


rdSt
idleSt
ackSt
wrSt
niosTrg
∧ rdCyc
niosTrg
∧ wrCyc
$\overline{\text{nCA}}$
$\overline{\text{nCA}}$
nCS

- Schreibzyklus


wrCyc
nCA
nCS
nWr
data
niosWr

`ioFsm` - Simulationsumgebung

- `tsb12lv01b.vhd` Simulationsmodell des Link-Layer ICs
  - Speicherschnittstelle
  - Datei Ein-/Ausgabe Textdatei für Stimuli / Speicherinhalte
  - Zähler für GRF-Werte optional
  - Timing-Checks Datenblatt
- `tsb12Pkg.vhd`
  - enthält Konstanten, Adressen, Maskierungen, Offsets
    für Link-Layer IC, FireWire Protokoll
- `ioFsmTB.vhd` Testumgebung
  - elementare Lese-/Schreiboperationen

`ioFsm` - Software- und Systemumgebung

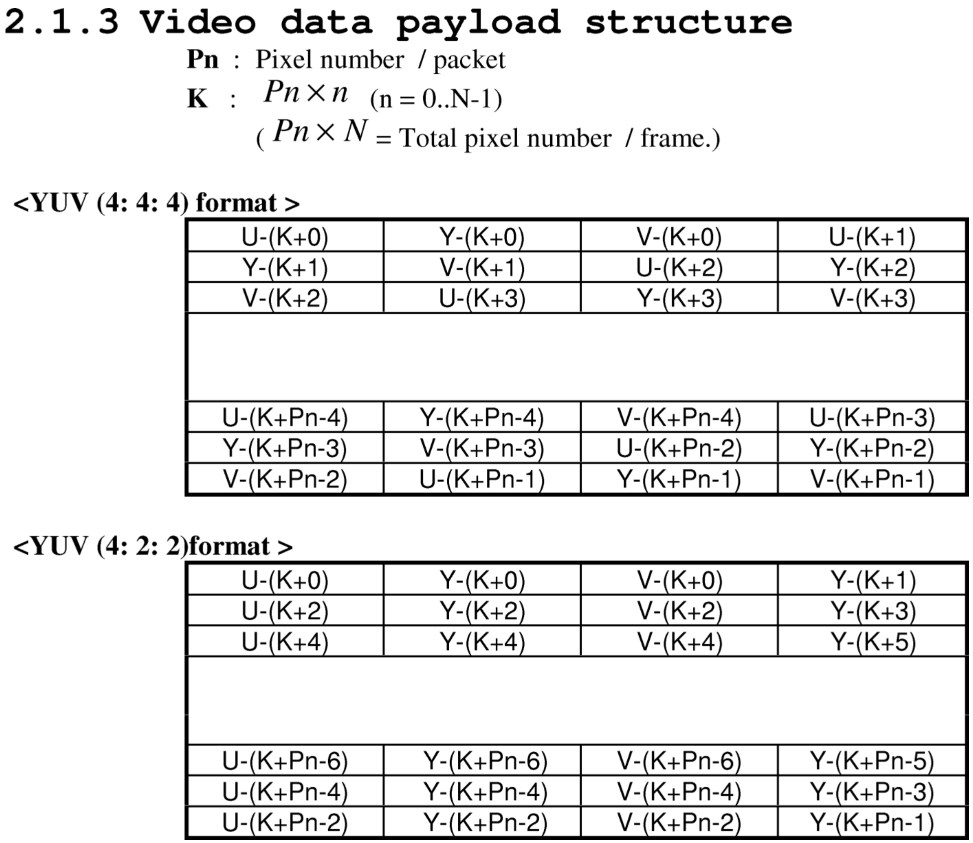
- `fw1394.h, fwTsb12.h, fwTsb41.h`
  - enthält Konstanten, Adressen, Maskierungen, Offsets
    für FireWire Protokoll, Link-Layer IC, Phys.-Layer IC
- `fwHWif.c` hardwarenahe Prozeduren
  - elementare Lese-/Schreiboperationen
  - Initialisierungs-Sequenz, Interrupt-Handling
- `fwRxData.c` Datenausgabe (-Verarbeitung)
  `fwUser.c` Benutzerschnittstelle
  `fwTest.c` Testprogramm
  - Zugriff auf die Hardware
  - FireWire Quadlets: Lesen/Schreiben

Vorgehensweise

- erster Prototyp wichtig `ioFsm` + `nios32` + Programm
  - Verhaltensweise/Eigenschaften des FireWire Protokolls und der Hardware
  - z.B.: Status (am Bus) erkennen
    Hardware-Initialisierung nach Bus-Reset
    ...
  - Anpassung der Software schnell, flexibel

Vorgehensweise

- Bearbeitung der Kameradaten
  - 640 · 480 Pixel
  - YUV(4:2:2)
  - 614400 byte/Bild
  - 153600 Quadlet
  - 240 Blöcke à 640 Q.
  - 30 fps
  - ≈140 Mbit/s



2.1.3 Video data payload structure

Pn : Pixel number / packet
K : $Pn \times n$ (n = 0..N-1)
($Pn \times N$ = Total pixel number / frame.)

<YUV (4: 4: 4) format >

U-(K+0)	Y-(K+0)	V-(K+0)	U-(K+1)
Y-(K+1)	V-(K+1)	U-(K+2)	Y-(K+2)
V-(K+2)	U-(K+3)	Y-(K+3)	V-(K+3)
U-(K+Pn-4)	Y-(K+Pn-4)	V-(K+Pn-4)	U-(K+Pn-3)
Y-(K+Pn-3)	V-(K+Pn-3)	U-(K+Pn-2)	Y-(K+Pn-2)
V-(K+Pn-2)	U-(K+Pn-1)	Y-(K+Pn-1)	V-(K+Pn-1)

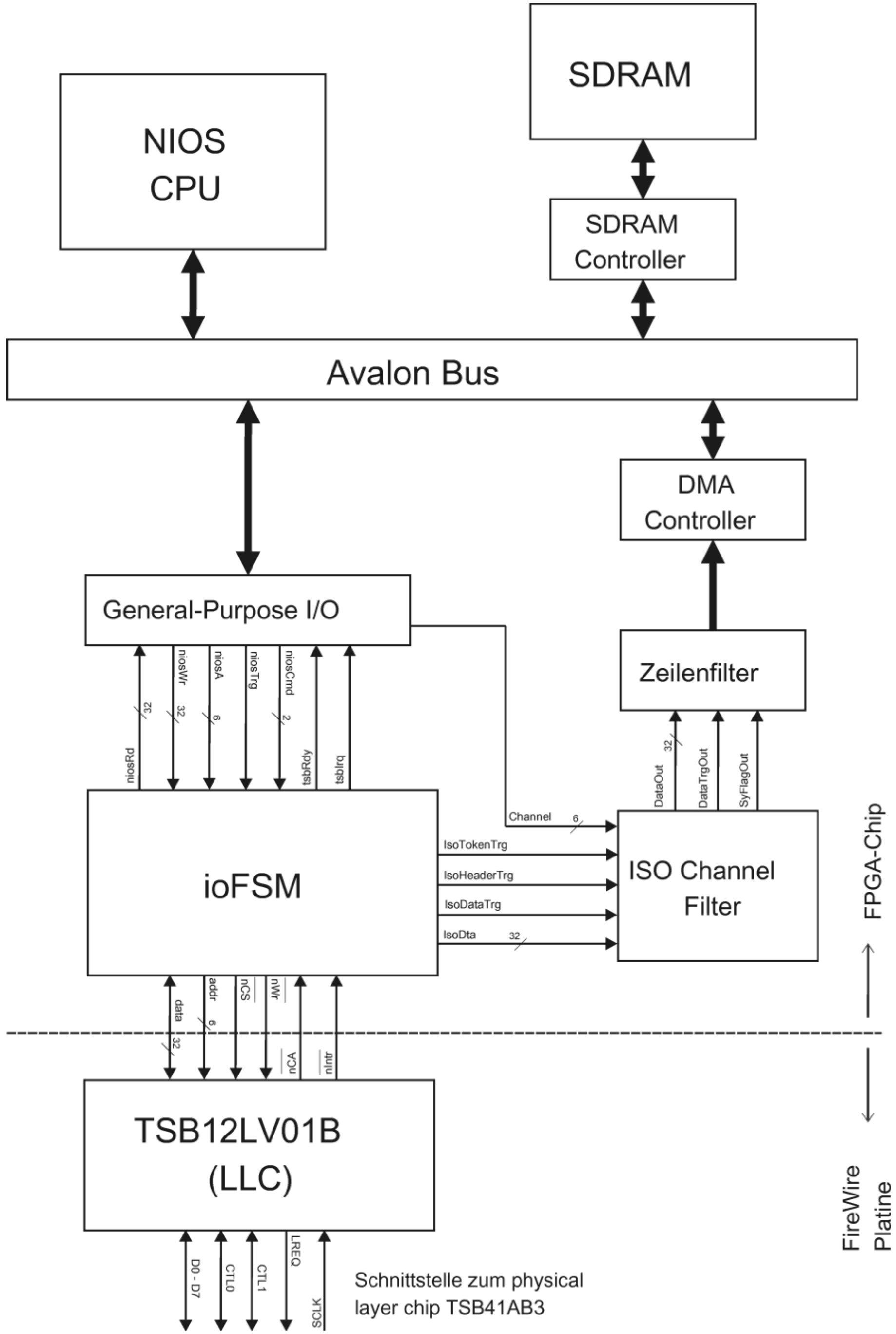
<YUV (4: 2: 2)format >

U-(K+0)	Y-(K+0)	V-(K+0)	Y-(K+1)
U-(K+2)	Y-(K+2)	V-(K+2)	Y-(K+3)
U-(K+4)	Y-(K+4)	V-(K+4)	Y-(K+5)
U-(K+Pn-6)	Y-(K+Pn-6)	V-(K+Pn-6)	Y-(K+Pn-5)
U-(K+Pn-4)	Y-(K+Pn-4)	V-(K+Pn-4)	Y-(K+Pn-3)
U-(K+Pn-2)	Y-(K+Pn-2)	V-(K+Pn-2)	Y-(K+Pn-1)

Vorgehensweise

- Hardware muss Daten abnehmen
  - Datenfilter trennt asynchrone und isochrone Daten
transparentes Prozessor-Interface für Async.
  - DMA-Komponente (Altera Nios IP)
transferiert isochr.-Daten in Speicher
- Verarbeitung durch Nios-CPU
- Probleme
  - Verarbeitungsleistung der CPU ⇒Bilder auslassen?
  - Speichergröße ⇒Bildausschnitte?

- Systemarchitektur


SDRAM
NIOS CPU
SDRAM Controller
Avalon Bus
DMA Controller
General-Purpose I/O
Zeilenfilter
niosRd
32
niosWr
32
niosA
6
niosTrg
niosCmd
2
tsbRdy
tsbIrq
DataOut
32
DataTrgOut
SyFlagOut
Channel
6
ioFSM
IsoTokenTrg
IsoHeaderTrg
IsoDataTrg
IsoDta
32
ISO Channel Filter
FPGA-Chip
data
32
addr
6
nCS
nWr
nCA
nIntr
TSB12LV01B (LLC)
FireWire Platine
D0 - D7
CTL0
CTL1
LREQ
SCLK

Schnittstelle zum physical layer chip TSB41AB3

Autofokus

- Testprogramme
  - Kamera (roh-) daten anzeigen — `rawView, filView`
  - Autofokus — `camView`
- Vereinfachungen
  - Grauwertbilder — y-Komponente
  - zentraler Bildausschnitt — 256 · 128 Pixel
  - Laplace-Filter — 3·3 Fenster
Integer-Arithmetik
  - Summe als Gütemaß

Autofokus

- „*quick and dirty*“ Verfahren

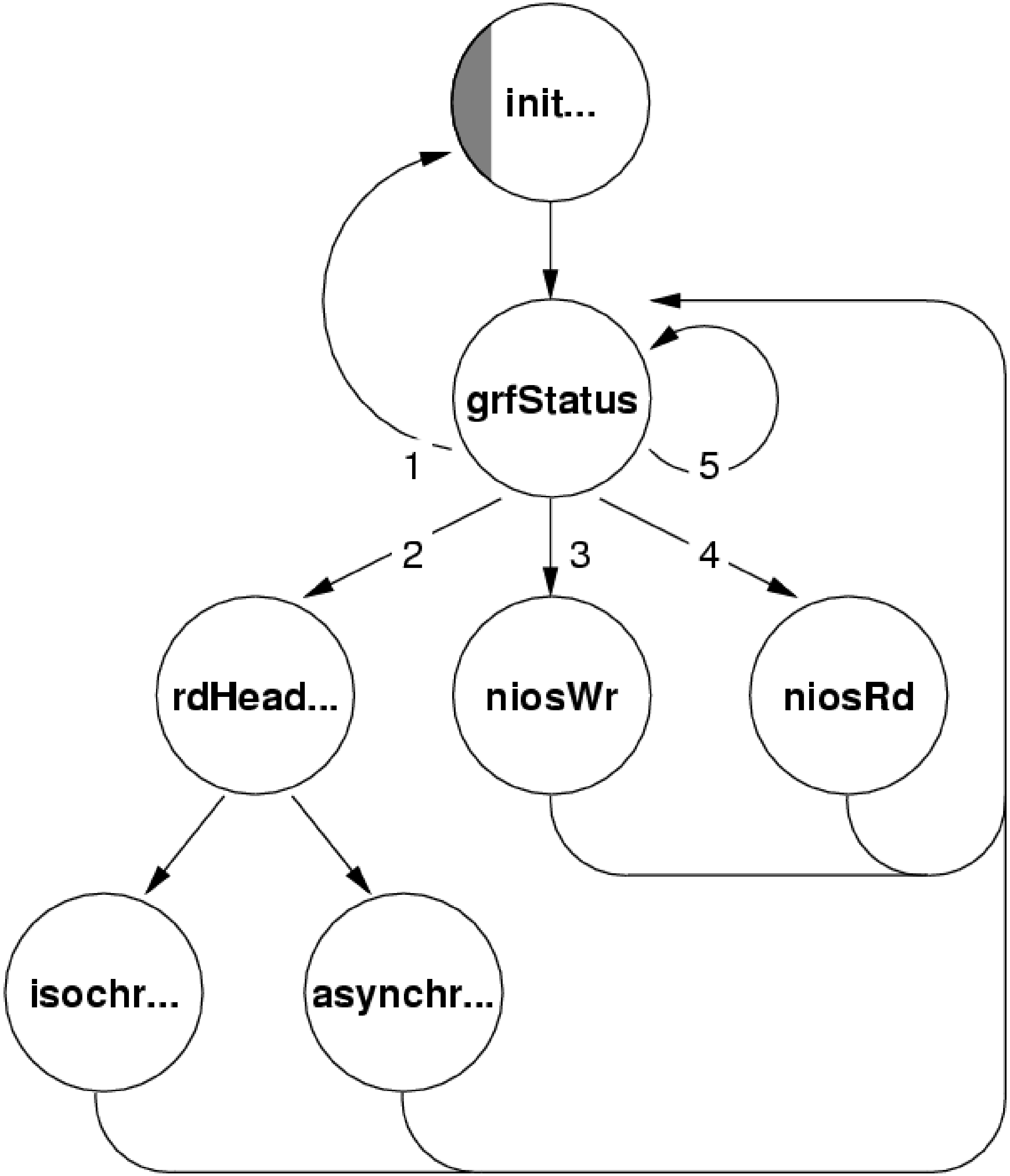
1. Intervallteilung	global
2. Gradientenverfahren	lokal
3. Schwellwert erzwingt Initialisierung	

- effizientere Methoden sollten folgen...

`reader`

- Aufgaben
    - holt Daten vom LLC-Chip *General Receive Fifo*
    - Trennung asynchroner- und isochroner Daten
    - Lese-/Schreibinterface zu CPU
    - Initialisierung der Hardware
      +Behandlung von Bus-Reset
- Implementation
    - Kontrollautomat
    - interner FIFO für asynchrone Daten 64 · 32 bit
    - CPU-Schnittstelle

reader


init...
grfStatus
1
5
2
3
4
rdHead...
niosWr
niosRd
isochr...
asynchr...

Initialisierung

zentrale Schleife

1 Interrupt (Bus-Reset)
2 Daten
3 CPU schreibt
4 " liest

erstes Quadlet lesen

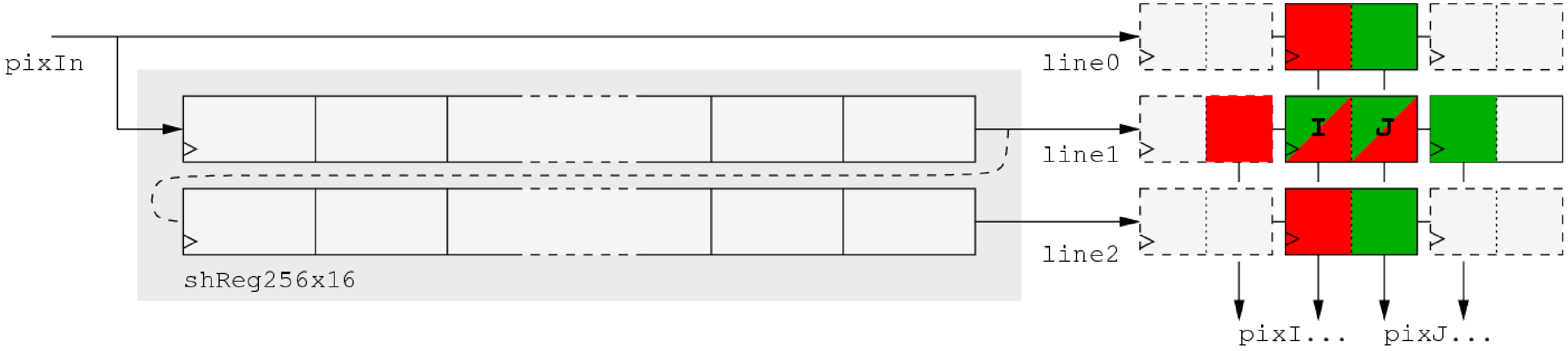
Burst-Lesezyklen

`filter`

- Berechnung des AF-Gütemaß als Hardware-Datenpfad
- *Reduktion* eines Bildes auf einen Wert (Merkmal)
- CPU (mit AF-Algorithmus)
  - erhält nur noch diese Werte
  - implementiert Regelung
  - schickt *neuen* Fokuswert an Kamera

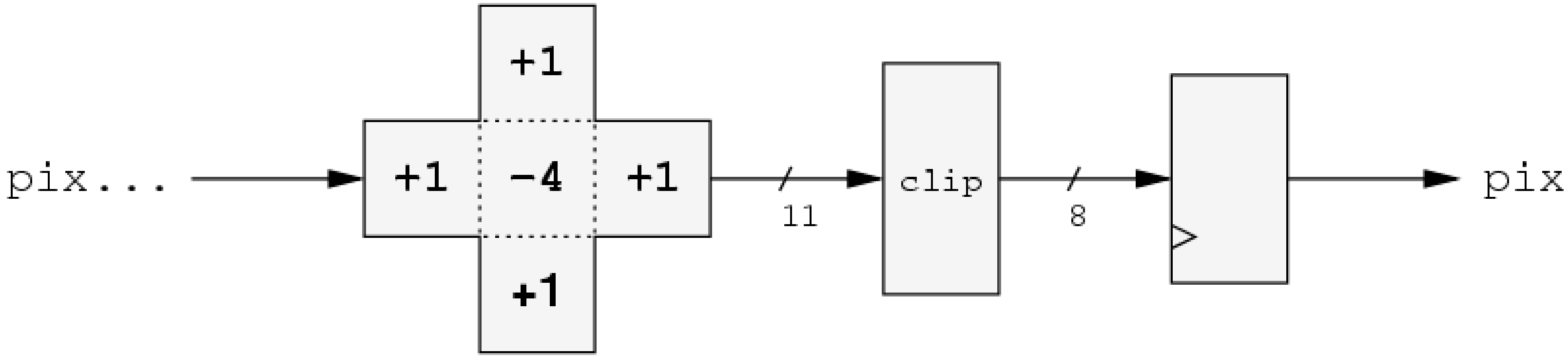
`filter` - Datenpfad

1. Delay-line und 3·3-Fensterung
   - jeweils 2 Pixel pro Takt YUV(4:2:2)
   - Schieberegister für 2-Bildzeilen


pixIn
line0
line1
line2
shReg256x16
I
J
pixI...
pixJ...

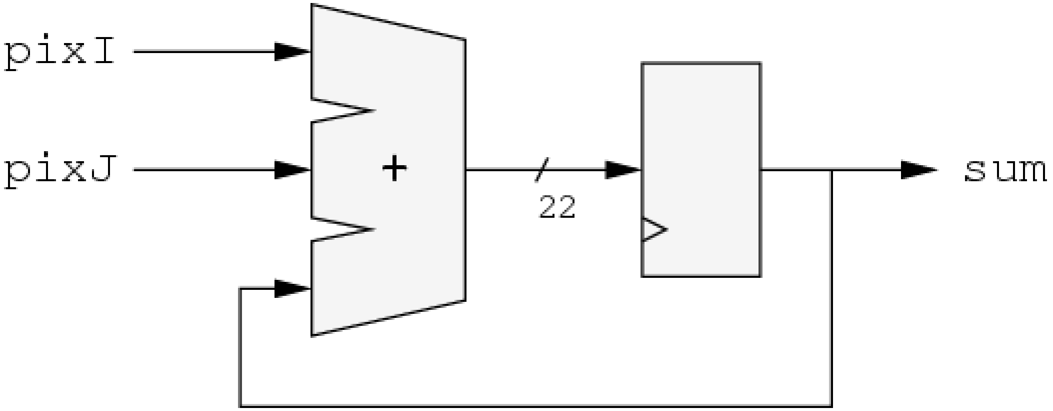
`filter` - Datenpfad

2. Laplace Filter
   - einfache Koeffizienten: 1, -4
   - Addition, Shift
   - Clipping



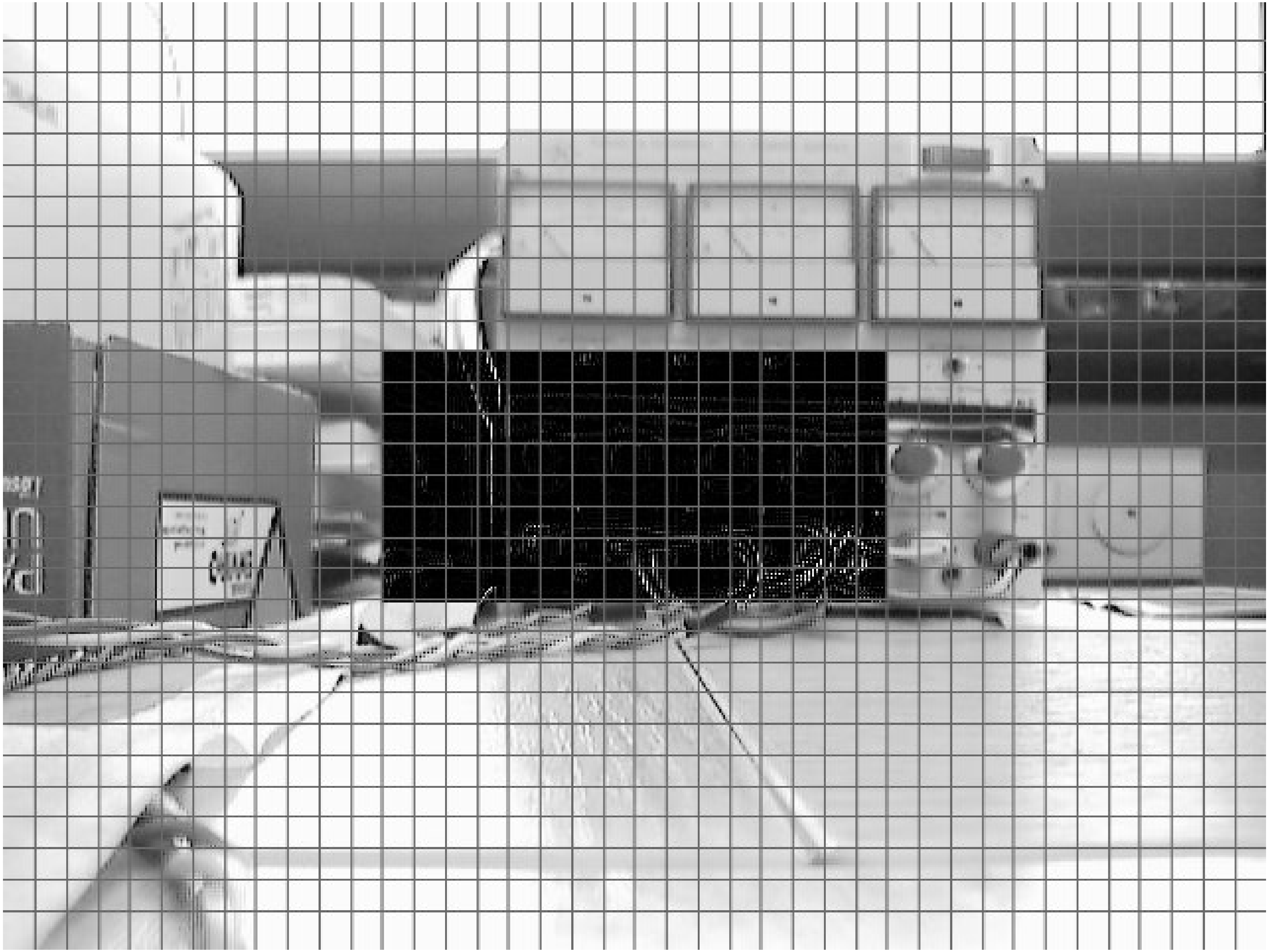
`filter` - Datenpfad

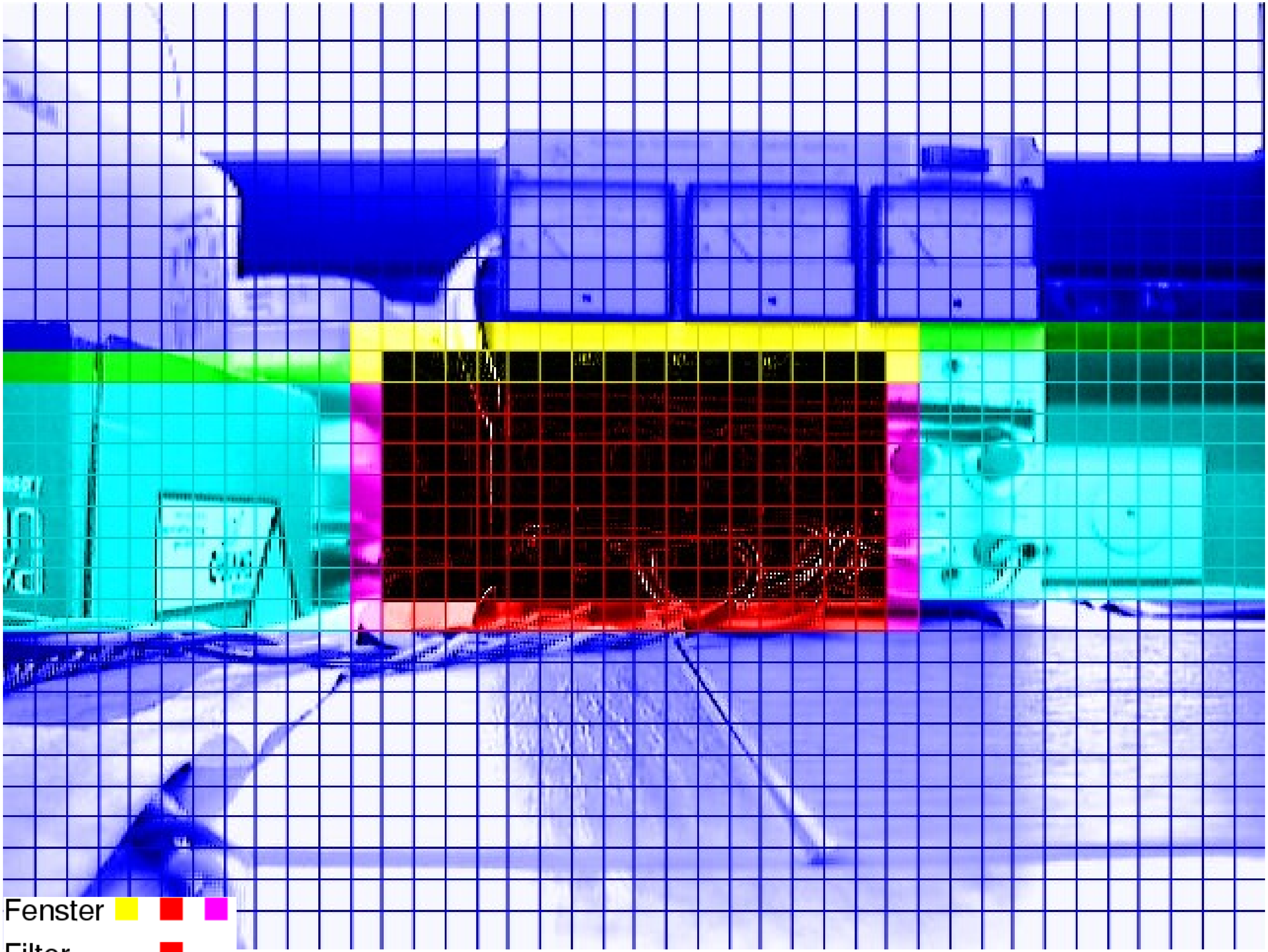
- Summe aller Filterwerte bilden



`filter` - Controller

- aktiviert die einzelnen Pipelinestufen *enable*-Leitungen
- Kamerabild
  - verteilt über mehrere isochrone Transfers
  - Kennungen Bildbeginn / Folgetransfer
  - Bildausschnitte ⇔ Zuständen des Automaten
    realisiert durch interne Zähler





Fenster 🟨 🟥 🟪

Filter 🟥

Summe ~Filter+1

`filter` - Simulationsumgebung

- `filterTB.vhd`	Testumgebung
- alternativ	
1. Zähler für Pixelwerte	prüft Pipeline-Timing
2. Dateieingabe	liest raw Bilddateien

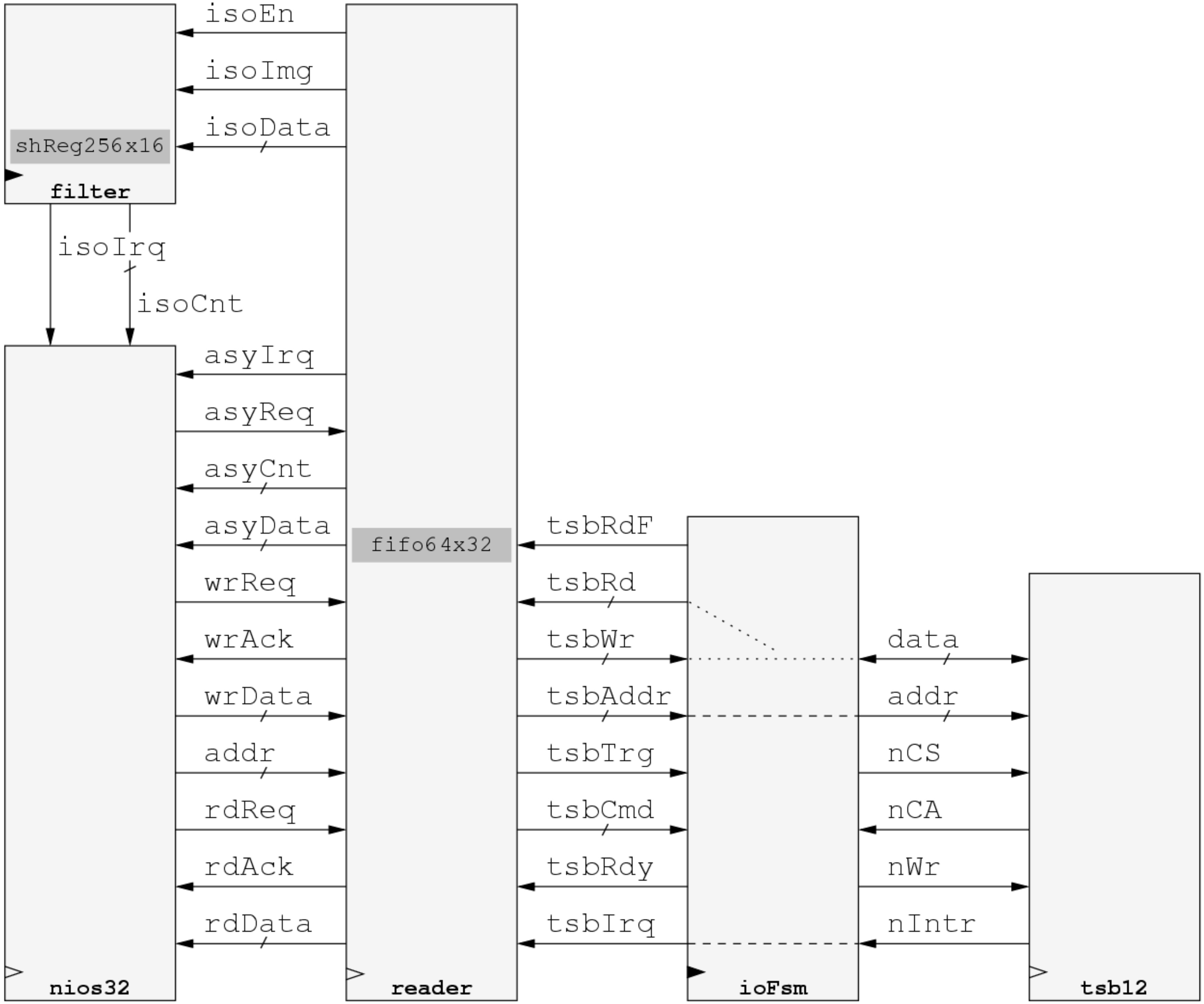
⇒ direkter Vergleich mit Software

system

- Aufbau des Gesamtsystems aus

`ioFsm`	Schnittstelle zu Low-Level Platine
`reader`	GRF-Daten abholen
`filter`	Datenpfad
`nios32`	CPU

system


isoEn
isoImg
isoData
shReg256x16
filter
isoIrq
isoCnt
asyIrq
asyReq
asyCnt
asyData
fifo64x32
wrReq
wrAck
wrData
addr
rdReq
rdAck
rdData
nios32
reader
tsbRdF
tsbRd
tsbWr
tsbAddr
tsbTrg
tsbCmd
tsbRdy
tsbIrq
ioFsm
data
addr
nCS
nCA
nWr
nIntr
tsb12

Probleme

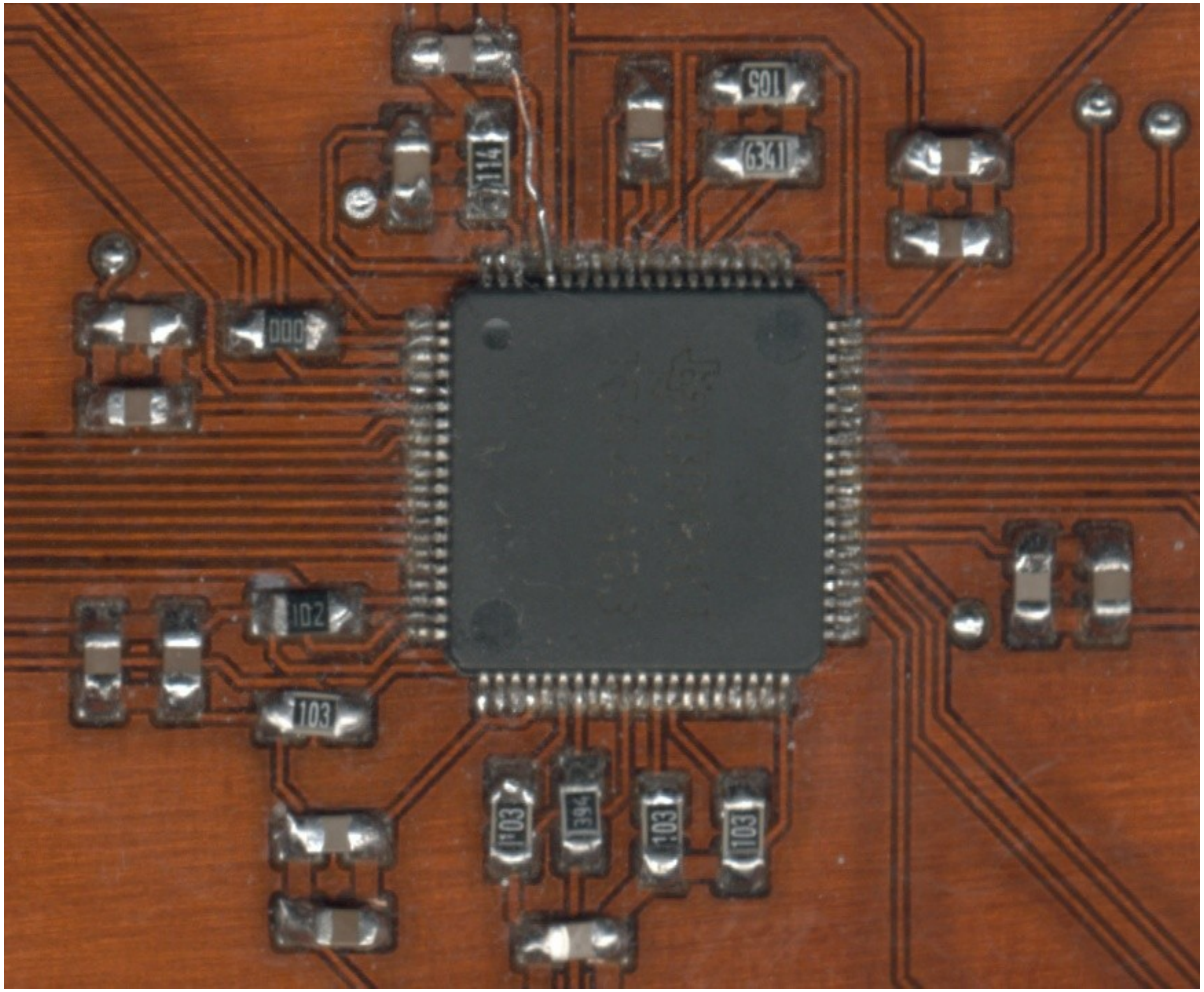
- Daten kommen falsch an ⇒ Platinenaufbau
- Kopplung der Automaten Vorder-/Rückflanke ⇒ Timing
- Reset der Nios-Platine ⇒ ?? Kurzschluss, Bus-Timing

⇒ einfacherer Aufbau `readFsm = ioFsm + reader`

aktuell: `llcFsm`

Probleme

- Platinenaufbau
- Lötstellen



Probleme

- Timing
  - *Clock-Routing* durch Nios-Board vorgegeben
  - PLL stellt Phase zu externer Clock ein `clkExt`

⇒ unsicher Flachbandkabel @33MHz
externe Verzögerungen

⇒ PLLs Jitter

- Takt des TI-Chips: 9...50 Mhz
  - Setup-/Hold-Zeiten erzwingen genau definiertes Timing

Probleme

- Lösung

⇒ Messung mit Oszilloskop / Timing-Analyzer
oft sind die Fehler nicht direkt ersichtlich!

⇒ Platinenverbindung

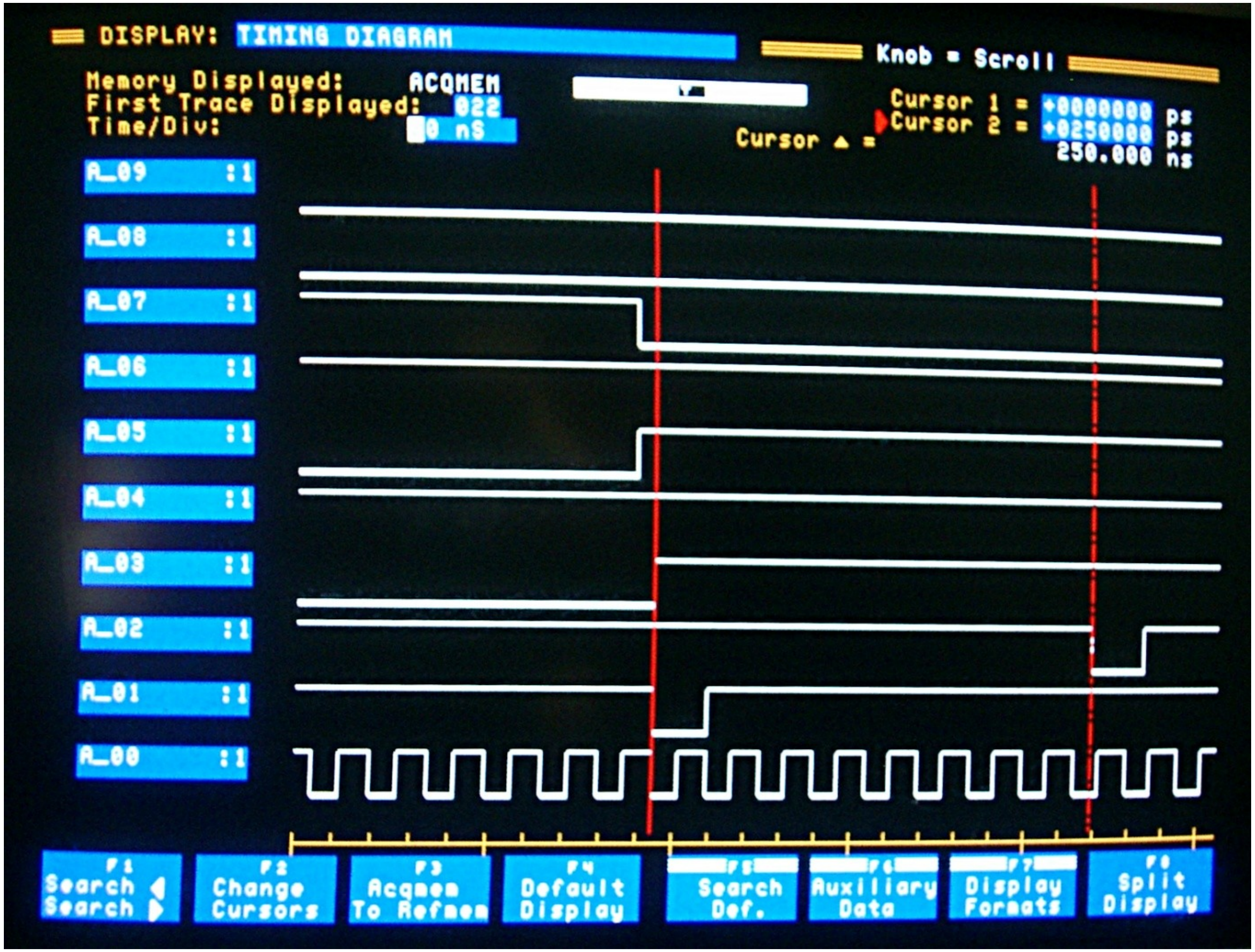
⇒ langsamerer Takt

Teiler:	2/3	22	MHz
	5/6	27.5	MHz

Timing

- 33 MHz
- *slow*-Cycle

→ sieht gut aus



DISPLAY: TIMING DIAGRAM

Memory Displayed: ACQMEM
First Trace Displayed: 022
Time/Div: 0 nS

Cursor 1 = +0000000 ps
Cursor 2 = +0250000 ps
Cursor ▲ = 250.000 ns

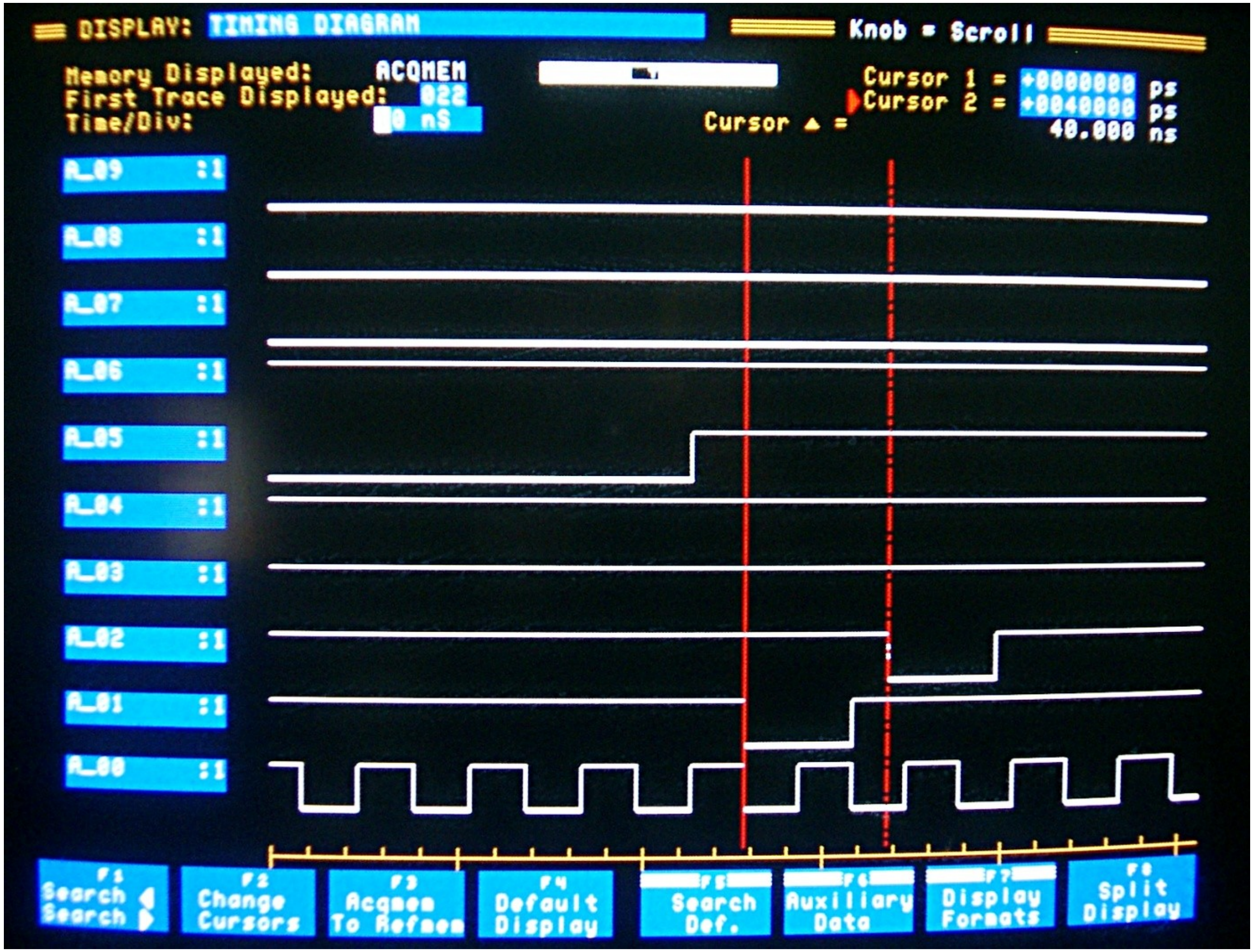
Knob = Scroll
A_09 :1
A_08 :1
A_07 :1
A_06 :1
A_05 :1
A_04 :1
A_03 :1
A_02 :1
A_01 :1
A_00 :1
F1 Search ◀ Search ▶
F2 Change Cursors
F3 Acqmem To Refmem
F4 Default Display
F5 Search Def.
F6 Auxiliary Data
F7 Display Formats
F8 Split Display

nCA
nCS
clk

Timing

- 33 MHz
- normaler Zugriff

→ Fehler
  - keine Reaktion in *einem* Takt
  - Software: OK
  - *Burst* Problem

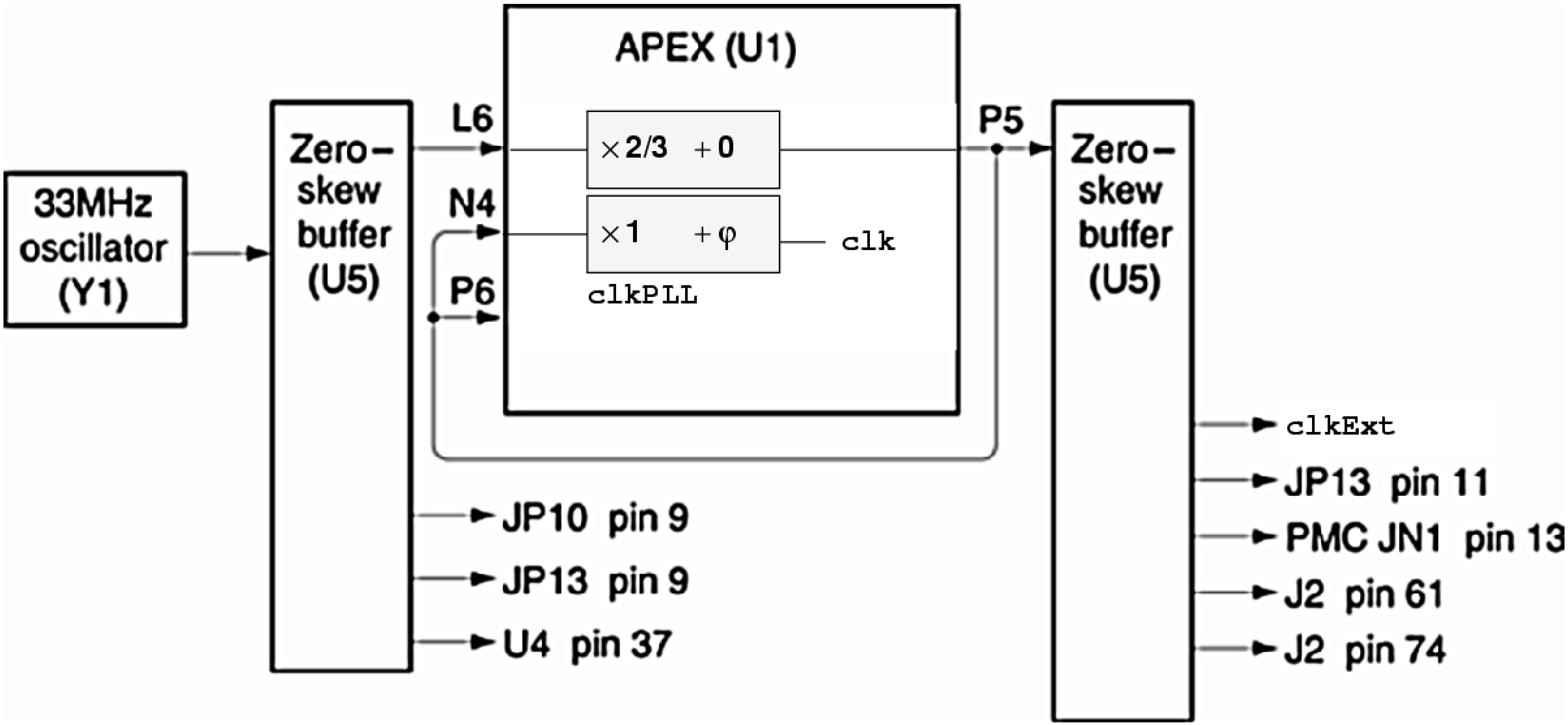


DISPLAY: TIMING DIAGRAM
Memory Displayed: ACQMEM
First Trace Displayed: 022
Time/Div: 0 nS
Cursor 1 = +0000000 ps
Cursor 2 = +0040000 ps
Cursor ▲ = 40.000 ns

Knob = Scroll
A_09 :1
A_08 :1
A_07 :1
A_06 :1
A_05 :1
A_04 :1
A_03 :1
A_02 :1
A_01 :1
A_00 :1
F1 Search ◀ Search ▶
F2 Change Cursors
F3 Acqmem To Refmem
F4 Default Display
F5 Search Def.
F6 Auxiliary Data
F7 Display Formats
F8 Split Display

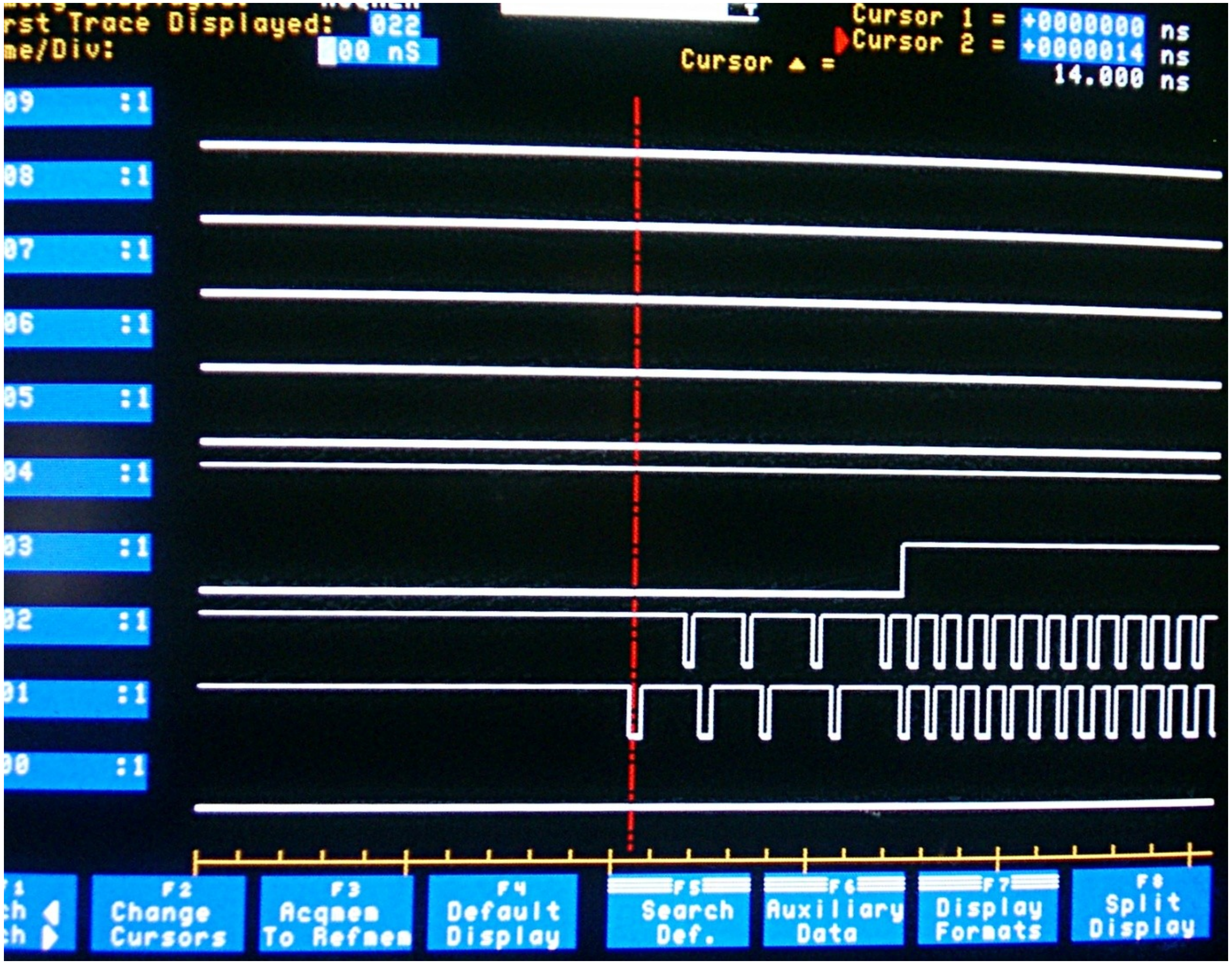
Taktgenerierung

- 22 MHz Taktung


33MHz oscillator (Y1)
Zero–skew buffer (U5)
L6
N4
P6
APEX (U1)
×2/3 +0
×1 +φ
clk
clkPLL
P5
Zero–skew buffer (U5)
clkExt
JP13 pin 11
PMC JN1 pin 13
J2 pin 61
J2 pin 74
JP10 pin 9
JP13 pin 9
U4 pin 37

Timing

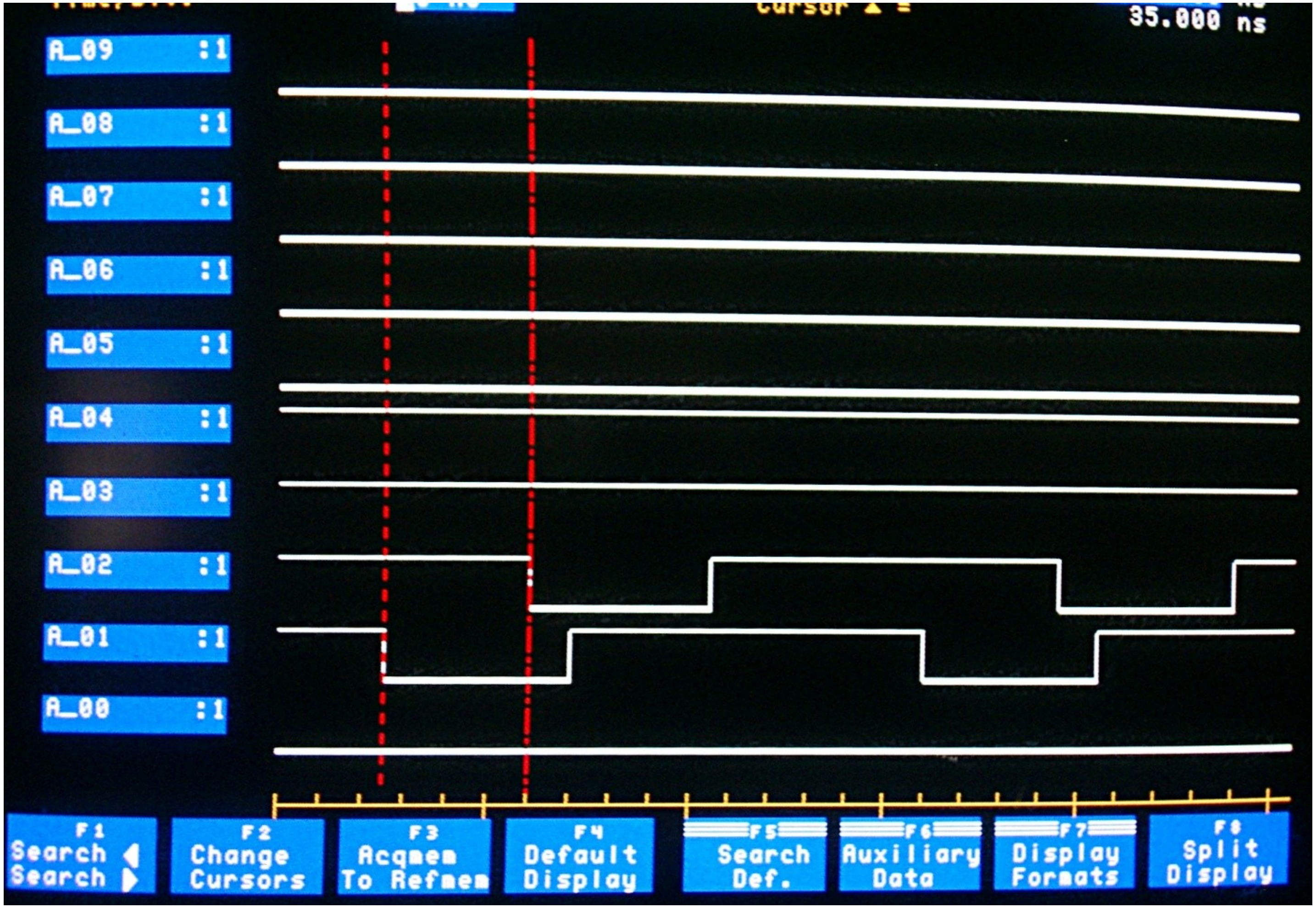
- 22 MHz
- `reader`-Init.


rst Trace Displayed: 022
me/Div: 00 nS
Cursor 1 = +0000000 ns
Cursor 2 = +0000014 ns
Cursor ▲ = 14.000 ns
09 :1
08 :1
07 :1
06 :1
05 :1
04 :1
03 :1
02 :1
01 :1
00 :1
Change Cursors
Acqmem To Refmem
Default Display
Search Def.
Auxiliary Data
Display Formats
Split Display

Timing

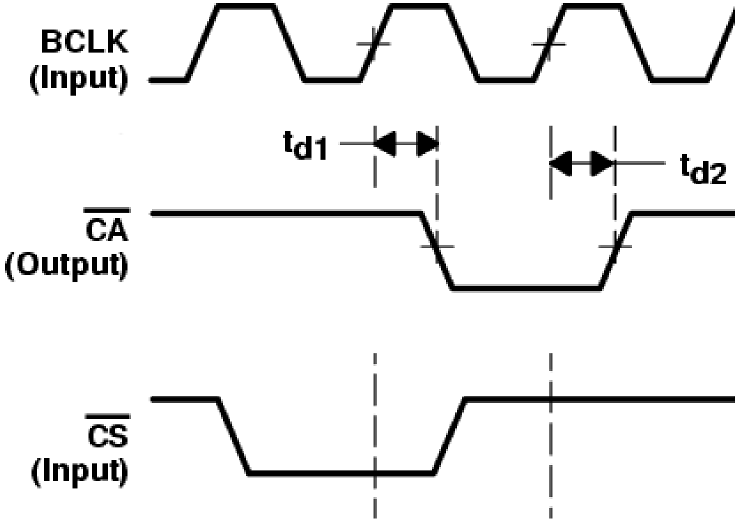
- 22 MHz

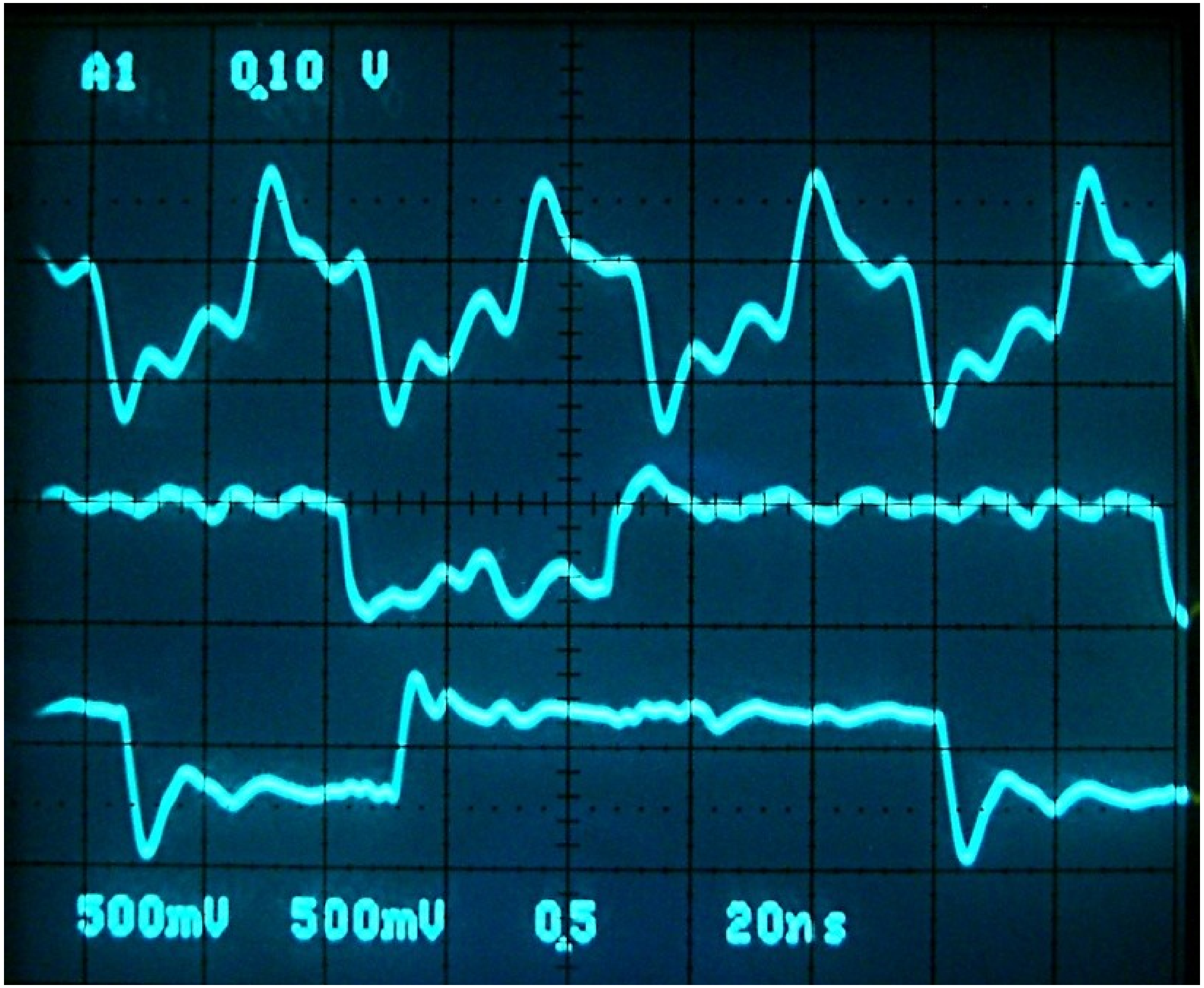
→ Datenblatt


Cursor ▲ =
35.000 ns
A_09 :1
A_08 :1
A_07 :1
A_06 :1
A_05 :1
A_04 :1
A_03 :1
A_02 :1
A_01 :1
A_00 :1
F1 Search ◀ Search ▶
F2 Change Cursors
F3 Acqmem To Refmem
F4 Default Display
F5 Search Def.
F6 Auxiliary Data
F7 Display Formats
F8 Split Display

Timing

- → Realität...


BCLK
(Input)
t_{d1}
t_{d2}
$\overline{\text{CA}}$
(Output)
$\overline{\text{CS}}$
(Input)



Fazit - Systementwurf

• Komplexität unterschätzt	Summe von Einzelproblemen
• systemnahe Simulation	
– Hardware	Bereitstellung der Modelle
– Software	Algorithmen
• Prototyp unverzichtbar	Unklarheiten der Spezifikation
• Probleme der *Hardware*	Platinenaufbau

- Anwendung (AF) zeigt Potenzial solcher eingebetteter Systeme

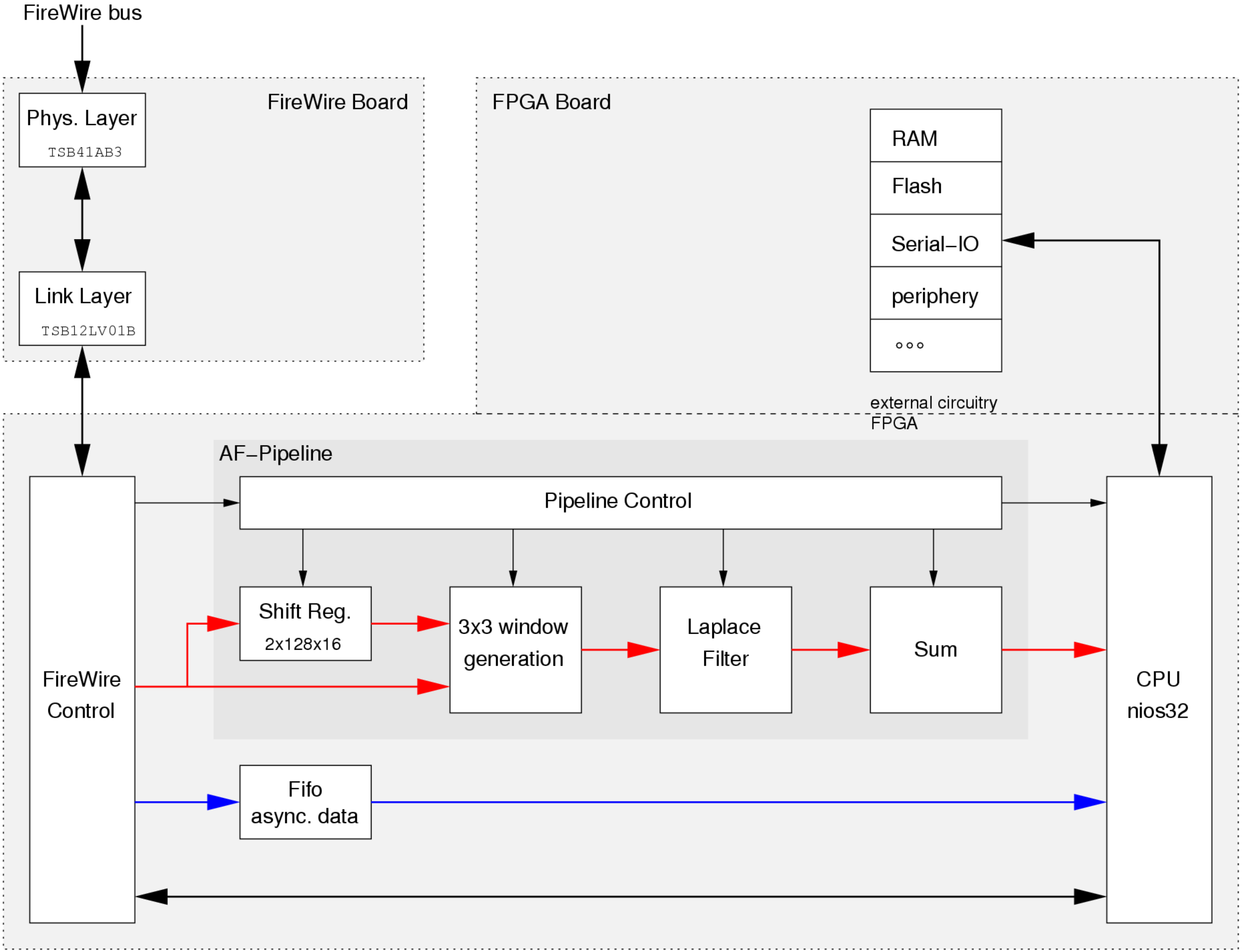
Fazit - Entwurfsvorgehen

- Schrittweise Entwicklung → auf Datenfluss bezogen

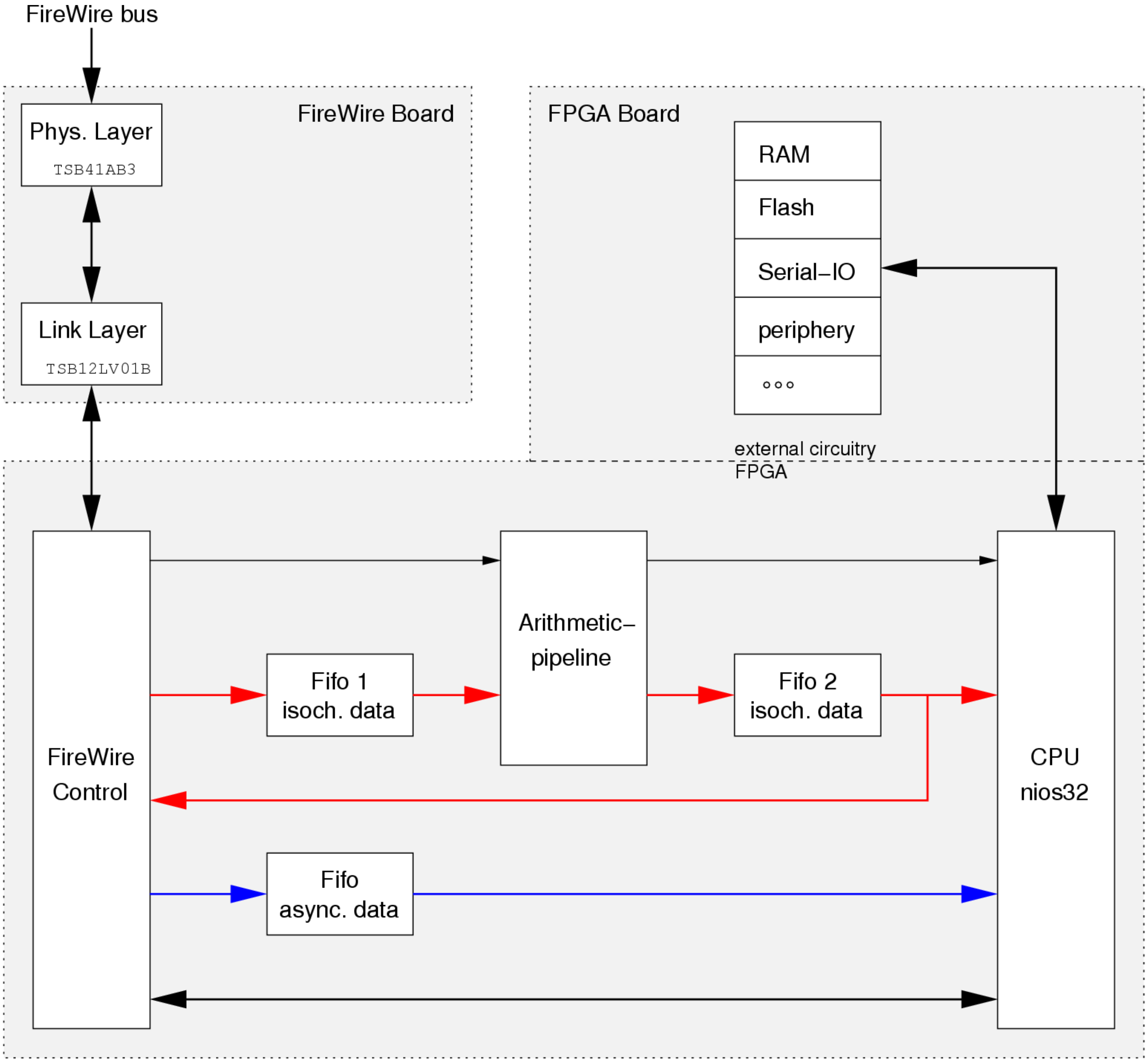
1. Kommunikation mit FireWire Bus
2. Isochrone Daten lesen ⇒ Autofokus
   Asynchrone -"- lesen+schreiben
   Schematic
3. Isochrone Daten lesen+schreiben ⇒ Filter, Egalisierung, ...
   Asynchrone -"- lesen+schreiben
   Schematic
4. Isochrone Daten lesen+schreiben ⇒ Omni. zu Panoramabild
   Asynchrone -"- lesen+schreiben
   SDRAM
   Schematic

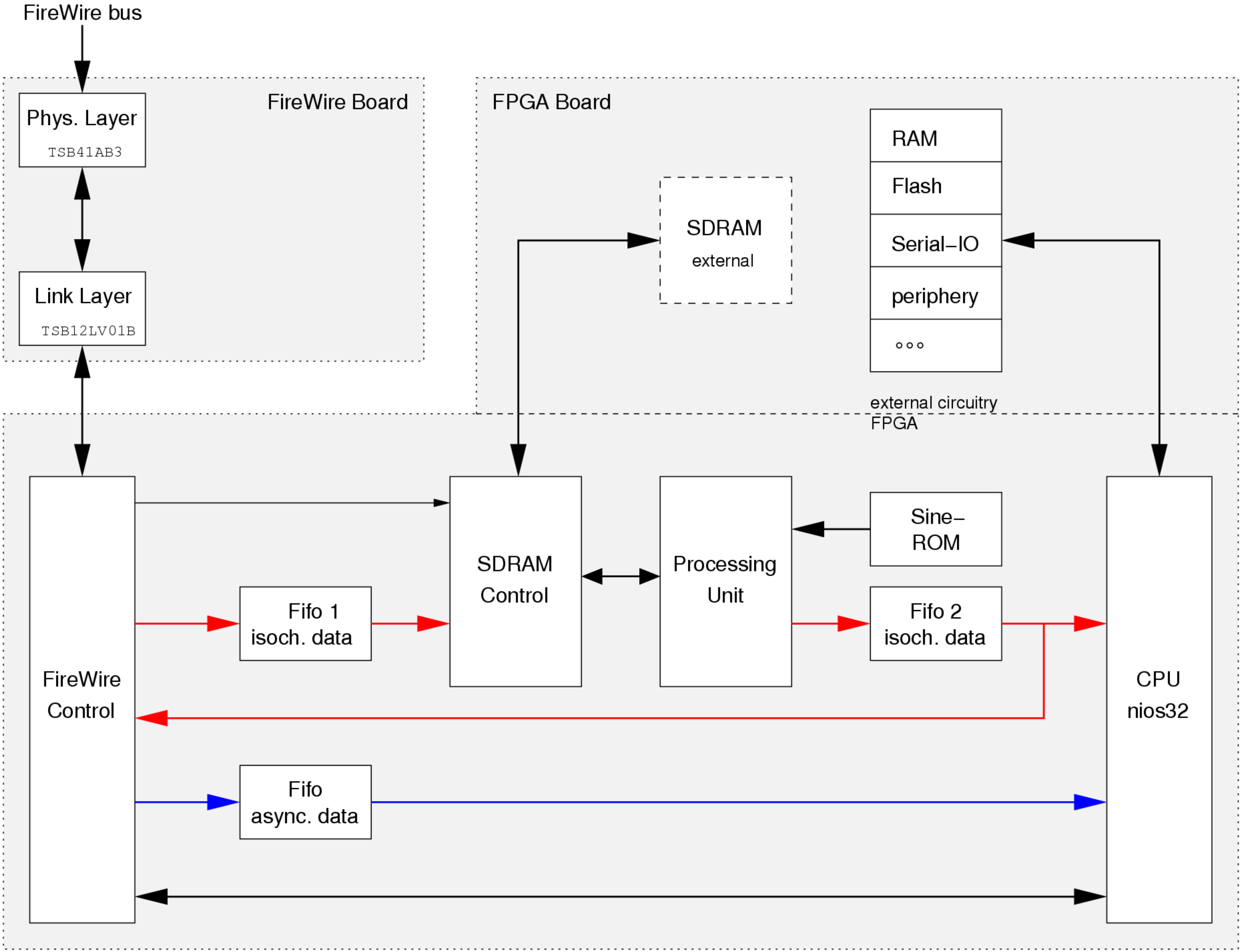
Fazit - Entwurfsvorgehen

- Schrittweise Entwicklung → auf Algorithmen bezogen
  1. Software auf PC
  2. Software auf eingebettetem Prozessor (NIOS)
  3. -"- + Hardwarekomponenten für zeitkritische Aufgaben
  4. Dedizierte Hardware
     (Kontrollfluss des Programms als Automat realisiert)



FireWire bus
FireWire Board
Phys. Layer
TSB41AB3
Link Layer
TSB12LV01B
FPGA Board
RAM
Flash
Serial–IO
periphery
∘∘∘
external circuitry
FPGA
AF–Pipeline
Pipeline Control
Shift Reg.
2x128x16
3x3 window generation
Laplace Filter
Sum
FireWire Control
CPU nios32
Fifo async. data


FireWire bus
FireWire Board
Phys. Layer
TSB41AB3
Link Layer
TSB12LV01B
FPGA Board
RAM
Flash
Serial–IO
periphery
◦◦◦
external circuitry
FPGA
FireWire Control
Fifo 1
isoch. data
Arithmetic–pipeline
Fifo 2
isoch. data
Fifo
async. data
CPU
nios32


FireWire bus
FireWire Board
Phys. Layer
TSB41AB3
Link Layer
TSB12LV01B
FPGA Board
SDRAM
external
RAM
Flash
Serial–IO
periphery
ooo
external circuitry
FPGA
FireWire
Control
Fifo 1
isoch. data
SDRAM
Control
Processing
Unit
Sine–
ROM
Fifo 2
isoch. data
CPU
nios32
Fifo
async. data