

# Praktikum angewandte Systemsoftwaretechnik

## Logic Analyzer

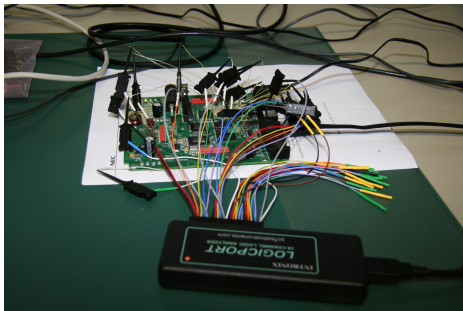
Alexander Würstlein

Lehrstuhl Informatik 4

2016-06-23

# Logic Analyzer

- Ein Logikanalysator wird verwendet um digitale Schaltungen zu analysieren (z.B. bei der Fehlersuche)
- Die Signale in einer Schaltung werden um ein im Voraus festgelegtes Ereignis herum aufgezeichnet (steigende oder fallende Taktflanke eines oder mehrerer Signale)



# Logic Analyzer – Ein (einfaches) PCI-Gerät

- Der in der Übung verwendete Logikanalysator wurde als Logik auf einer PCI FPGA-Karte realisiert
- Der Analysator hat 16 Signaleingänge, welche fest mit einem Signalgenerator verbunden sind
- Das Gerät bietet zwei Speicherbereiche (*memory mapped register*)
  - Region 0: Kommunikation mit der PCI-Schnittstelle des FPGA
  - Region 1: Kommunikation mit dem Logikanalysator
- Die Karte schreibt aufgezeichnete Daten über DMA in den Speicher
- Über einen Interrupt wird das vollständige Beschreiben einer Speicherseite mit aufgezeichneten Daten signalisiert

```
# lspci -vvv
02:09.0 System peripheral: Device 4242:2323 (rev 01)
  Subsystem: Device 4242:2323
  Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- ...
  Status: Cap- 66MHz- UDF- FastB2B+ ParErr- DEVSEL=medium >TAbort- <TAbort- ...
  Latency: 66 (2000ns min, 6500ns max), Cache Line Size: 32 bytes
  Interrupt: pin A routed to IRQ 11
  Region 0: Memory at d0109000 (32-bit, non-prefetchable) [size=4K]
  Region 1: Memory at d0100000 (32-bit, non-prefetchable) [size=8K]
```

# Logic Analyzer – Register

- Konfiguration des Geräts geschieht über Register
  - In den Kernel-Speicher eingeblendet
  - im Userspace auch: `/sys/bus/pci/devices/<id>/resouce<num>`
  - Jeweils 32 Bit breit
  - Zugriff mit `readl()` und `writel()`

Region	Adresse	Name	Beschreibung
0	0x1e8	LA_CFG_IACK	Interruptbestätigung PCI-Int.
0	0x1ec	LA_CFG_ICR	Interruptkonfiguration
0	0x1f0	LA_CFG_ISR	Interruptstatus
1	0x000	LA_STATUS	Statuswort
1	0x008	LA_COMMAND	Kommando
1	0x00c	LA_TRIGGER_RISING	Trigger auf steigende Flanken
1	0x01c	LA_TRIGGER_FALLING	Trigger auf fallende Flanken
1	0x02c	LA_SAMPLE_RATE	Abtastrate
1	0x034	LA_MEMORY_PAGE	Speicherseiten für DMA
1	0x0fc	LA_ISR, LA_IACK	Interruptstatus/-bestätigung LA

Header mit Definitionen: `/proj/i4passt/la/la.h`

## Logic Analyzer – Interrupts

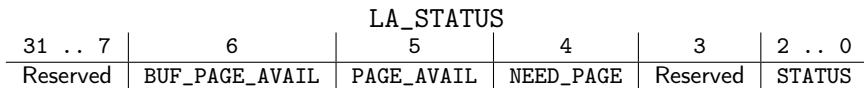
- LA\_CFG\_ICR: In diesem Register werden die Interruptquellen des PCI-Geräts an- und abgeschaltet. Der LA-Interrupt kann mit dem Wert 0x01 aktiviert und mit dem Wert 0x00 deaktiviert werden.
- LA\_CFG\_ISR: Wenn dieses Register ungleich 0x00 ist, hat das Gerät einen (evtl. mit mehreren Geräten geteilten) Interrupt ausgelöst. Sollte dies der Fall sein, muss das Register in der Unterbrechungsbehandlung auf 0x00 zurückgesetzt werden.
- LA\_CFG\_IACK: Durch **lesen** dieses Registers wird das Interruptsignal des Geräts bestätigt und dieses nimmt das Signal von der Interruptleitung zurück.

## Logic Analyzer – Interrupts (2)

- LA\_ISR, LA\_IACK: Ein **lesender** Zugriff auf dieses Register gibt den Unterbrechungsgrund des LA an (immer 0x02), ein **schreibender** Zugriff dieses Wertes bestätigt diesen.
- Da der Logikanalysator aus zwei Einheiten aufgebaut ist (PCI-Schnittstelle und LA-Logik), müssen Unterbrechungen an beiden Einheiten bestätigt werden.

```
#define LA_PCI_INT_LA      (1<<0)
#define LA_PCI_INT_WBE    (1<<1)
#define LA_PCI_INT_PCIE   (1<<2)
#define LA_PCI_INT_PERR   (1<<3)
#define LA_PCI_INT_SERR   (1<<4)
```

# Logic Analyzer – Statuswort



- BUF\_PAGE\_AVAIL: Eine DMA-Seite auf Vorrat ist verfügbar
- PAGE\_AVAIL: Eine DMA-Seite für aktuelle Daten ist verfügbar
- NEED\_PAGE: Eine zusätzliche DMA-Seite wird benötigt
- LA\_STATUS: Aktueller Zustand des Logikanalysators:
  - 0x00: Initialisierungsphase
  - 0x01: Auf Trigger wartend
  - 0x02: Aufnehmend
  - 0x03: Angehalten

```
#define LA_STATUS_INIT          0x00
#define LA_STATUS_WAITING      0x01
#define LA_STATUS_RECORDING    0x02
#define LA_STATUS_STOPPED      0x03
```

# Logic Analyzer – Kommando

LA_COMMAND	
31 .. 2	1 .. 0
Reserved	COMMAND

- **COMMAND:** Kommando an den Logikanalysator:
  - 0x02: Auf Triggerereignis warten und danach mit der Aufzeichnung beginnen. Dieses Kommando wird nur akzeptiert, wenn der LA angehalten ist.
  - 0x03: Den Logikanalysator aus jedem Zustand anhalten.

```
#define LA_CMD_ARM_TRIGGER      0x02
#define LA_CMD_STOP            0x03
```



# Logic Analyzer – Trigger und Abtastrate

LA_TRIGGER_{RISING,FALLING}	
31 .. 16	15 .. 0
Reserved	TRIGGER_BITS

- TRIGGER\_BITS: Die Aufzeichnung wird gestartet, wenn ein zu einem gesetzten Bit gehörendes Signal die entsprechende Zustandsänderung durchführt. Die Signale sind hierbei ODER-verknüpft (0xffff triggered, sobald ein Signal den Zustand entsprechend ändert).

LA_SAMPLE_RATE
31 .. 0
SAMPLE_RATE

- SAMPLE\_RATE: Teiler der Basisfrequenz (33MHz) für die Abtastrate. Nicht jeder Wert kann eingestellt werden, ein **lesender** Zugriff gibt den wirklich eingestellten Teiler aus.

```
#define BASE_FREQ 33000000
```

# Logic Analyzer – Speicherseiten für DMA

$$\frac{\text{LA\_MEMORY\_PAGE}}{\text{31 .. 0}} \\ \text{MEMORY\_PAGE}$$

- **MEMORY\_PAGE:** Wenn das **NEED\_PAGE**-Bit im Statuswort gesetzt ist, kann in dieses Register die physikalische Adresse einer DMA-Speicherseite geschrieben werden. Wenn ein Interrupt ausgelöst wurde und dieses Register beim Lesen einen Wert ungleich 0x00 enthält, so ist dies die physikalische Adresse der soeben mit abgetasteten Daten gefüllten Seite.
- Das Gerät kann mehr als eine DMA-Speicherseite vorhalten, deshalb sollte das **NEED\_PAGE**-Bit nach nach jedem Schreiben in **MEMORY\_PAGE** erneut überprüft werden.

## Logic Analyzer – Speicherseiten für DMA

- Das Gerät beschreibt jede Seite nur einmal, d.h. es „verbraucht“ die ihm übergebenen Speicherseiten.
- Soll eine bereits beschriebene DMA-Speicherseite nochmals verwendet werden so muss diese vom Treiber nochmals an die Karte übergeben werden.

## Logic Analyzer – Initialisierung

- Registrierung eines Treibers für ein PCI-Gerät mit der *Vendor-ID* 0x4242 und der *Product-ID* 0x2323

```
DEFINE_PCI_DEVICE_TABLE(passt_pci_tbl) = {  
    { PCI_DEVICE(0x4242, 0x2323) },  
    { 0 }  
};
```

- Aktivierung des PCI-Geräts (`pci_enable_device(dev)`)
- Konfiguration der PCI-Eigenschaften:

```
u16 pci_command;  
pci_read_config_word(dev, PCI_COMMAND, &pci_command);  
pci_command |= PCI_COMMAND_MASTER | PCI_COMMAND_MEMORY |  
               PCI_COMMAND_IO | PCI_COMMAND_SERR;  
pci_write_config_word(dev, PCI_COMMAND, pci_command);
```

## Logic Analyzer – Initialisierung (2)

- Speicherbereiche (BAR<sup>1</sup>) zuweisen (0 = PCI, 1 = LA):

```
pci_bar = pci_ioremap_bar(dev, 0);  
pci_addr = pci_resource_start(dev, 0);  
pci_size = pci_resource_len(dev, 0);
```

- Unterbrechung anfordern und Behandlungsfunktion registrieren:

```
request_irq(..., IRQF_SHARED, ...);
```

- Unterbrechungen durch schreiben in Register aktivieren:

```
writel(0x01, pci_bar + LA_CFG_ICR); /* LA_PCI_INT_LA = 0x01 */
```

- Als Dokumentation bietet sich das Buch „*Linux Device Drivers*”<sup>2</sup> an

---

<sup>1</sup>Base Address Register

<sup>2</sup><http://lwn.net/Kernel/LDD3/>, Kapitel 9 & 12

# Logic Analyzer – Abtasten

- Triggermasken setzen
- Abtastrate in Teiler umrechnen und setzen
- DMA-Speicherseiten anfordern und dem Gerät zuweisen, solange das `NEED_PAGE`-Bit gesetzt ist
- Das Kommando `LA_CMD_ARM_TRIGGER` absetzen
- Auf Interrupts warten
- Bei Eintreffen eines Interrupts:
  - Interruptquelle überprüfen
  - Unterbrechung am PCI- und LA-Gerät bestätigen
  - Beschriebene Speicherseite merken
  - DMA-Speicherseiten anfordern und dem Gerät zuweisen, solange das `NEED_PAGE`-Bit gesetzt ist
  - Unterbrechungsbehandlung beenden
- Beschriebene Speicherseiten über eine Gerätedatei in den Userspace kopieren (`open`, `read`, `poll`)

# Logic Analyzer – DMA

- Abgetastete Daten werden per DMA in den Hauptspeicher geschrieben
- Es wird immer eine Speicherseite (4096 Bytes) beschrieben
- Die physikalische Adresse einer DMA-fähigen Speicherseite muss an das Gerät übergeben werden
- Anforderung einer DMA-Seite durch `dma_alloc_coherent` oder einen DMA-Pool
  - DMA-Funktionen liefern virtuelle und physikalische Adresse der Speicherseite
  - Dokumentation in `linux/Documentation/DMA-API.txt`

# Logic Analyzer – RLE

- Abgetastete Daten werden mittels Lauflängenkodierung komprimiert
  - Datenmenge über den PCI-Bus wird reduziert  
⇒ Ermöglicht eine höhere Abtastrate
- Ein Sample des Logikanalysators ist 32 Bit lang

31 .. 16		15 .. 0
COUNT		SIGNAL

- COUNT: Anzahl der Takte, die das Signal unverändert geblieben ist
- SIGNAL: Datensignale

```
$ xxd /dev/la
0000000: 0004 XXXX 0004 XXXY 0005 YYYY 0004 ZZZZ  ....
```



# Logic Analyzer – Datensignal und Ausgabe

- Der Logikanalysator ist fest mit einem Signalgenerator verbunden
- Der Generator erzeugt zwei unabhängige 8 bit Signale
- Die Frequenz der erzeugten Signale beträgt ungefähr 2MHz (d.h. min. 4 MHz Abtastrate nötig)
- Das dekodierte Signal soll in eine Textdatei geschrieben werden, wobei Signale zu Bussen in unterschiedlichen Formaten (Hex und ASCII) zusammengefasst werden sollen. Eine Zeile soll einem Takt entsprechen:

```
$ cat la.out
15..8 7 6 5 4 3..0
  H 0 1 0 1 0xa
  a 1 0 1 1 0xf
  l 1 1 0 1 0xf
  l 1 1 1 1 0xe
  o 0 0 0 0 0x0
```

- Wie lautet der im Signal kodierte geheime Text?

# Logic Analyzer – Aufgabe

- Entwicklung eines Treibers
  - Registerzugriffe
  - Unterbrechungsbehandlung
  - DMA
  - Geräteoperationen
- Entwicklung einer Anwendung für den Logikanalysator
  - Konfiguration der Abtastparameter
  - Aufbereitung der abgetasteten Daten
- Entwicklungsumgebung: Rechner *i4passt{0,1,2}.informatik...*
  - Keine virtuelle Maschine
  - Root-Passwort wird bekanntgegeben
  - Sourcen zum lfd. Kernel in `/usr/src`
- Abgabe: 2016-07-14