Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

ITIV

Hardware/Software Co-Design

Übung 4 - Wiederholung

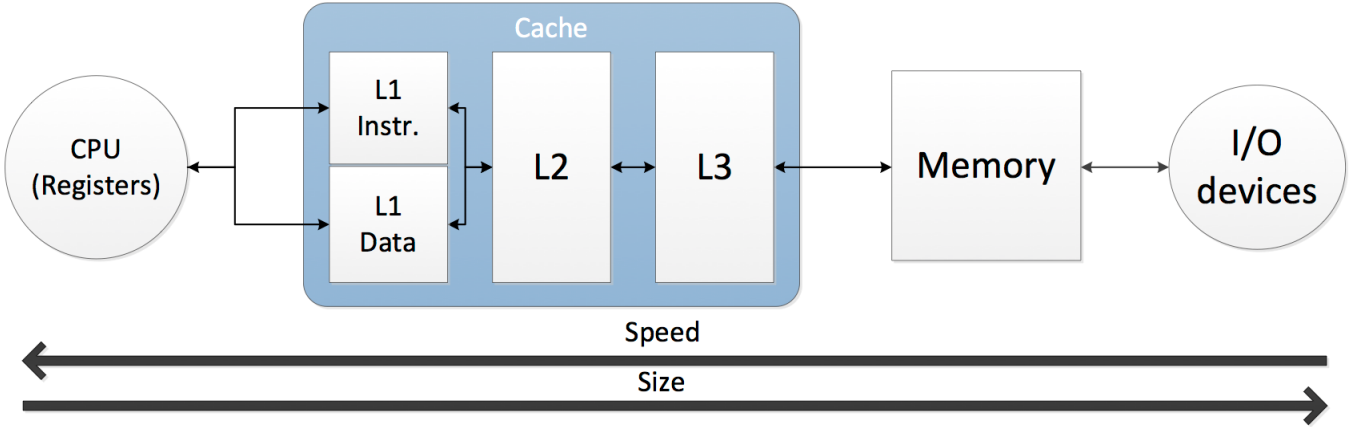
M.Sc. Fabian Lesniak

Agenda

- Wiederholung ausgewählter Themen
  - Caches
  - µC/DSP Begriffe
  - FPGA

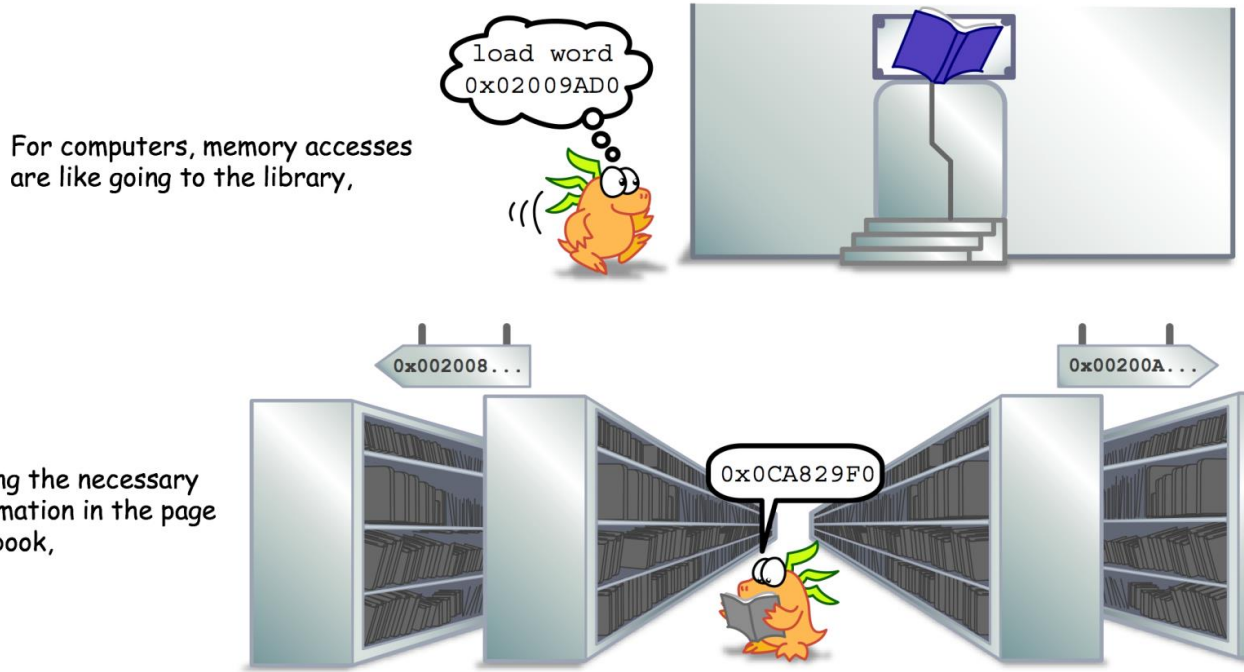
2.3.2.5 Caches - Hierarchie

- In der Regel besitzt ein Rechner einen getrennten Cache für Instruktionen (Instruktionscache) und für Daten (Datencache)
- Cache Level: Level 1 bis Level N bezeichnet die Cache Speicher beginnend mit dem schnellsten Speicherzugriff



Bachelorarbeit Maximilian Braun, ITIV 2012

2.3.2.5 Caches

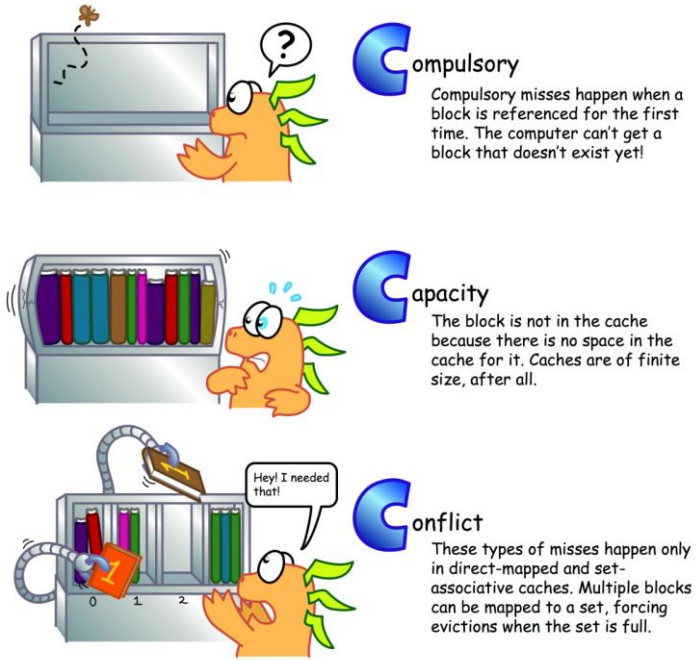


http://csillustrated.berkeley.edu/PDFs/handouts/

2.3.2.5 Caches

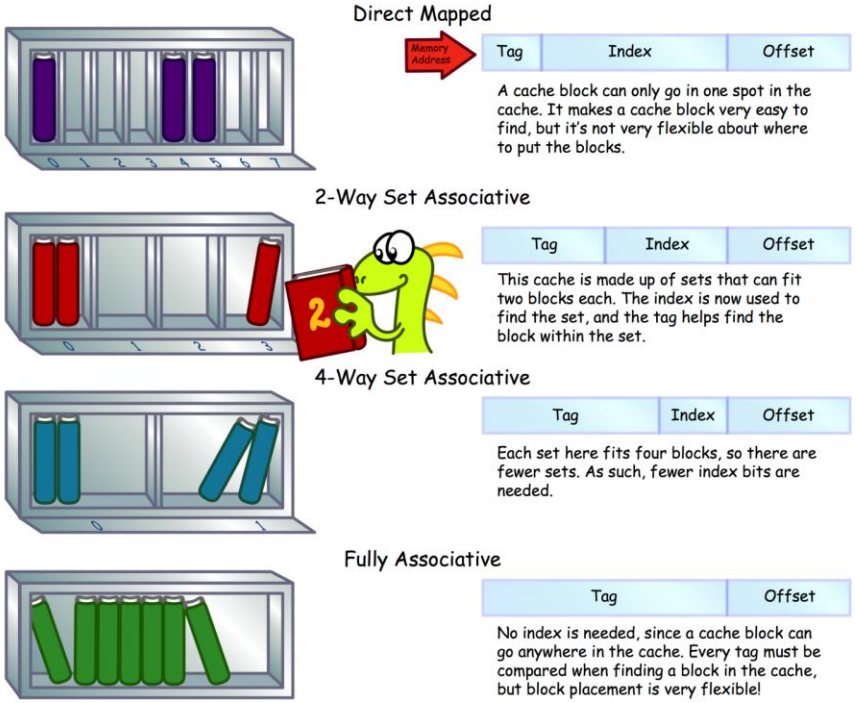
http://csillustrated.berkeley.edu/PDFs/handouts/

2.3.2.5 Cache Misses



http://csillustrated.berkeley.edu/PDFs/handouts/

2.3.2.5 Cache Assoziativität



http://csillustrated.berkeley.edu/PDFs/handouts/

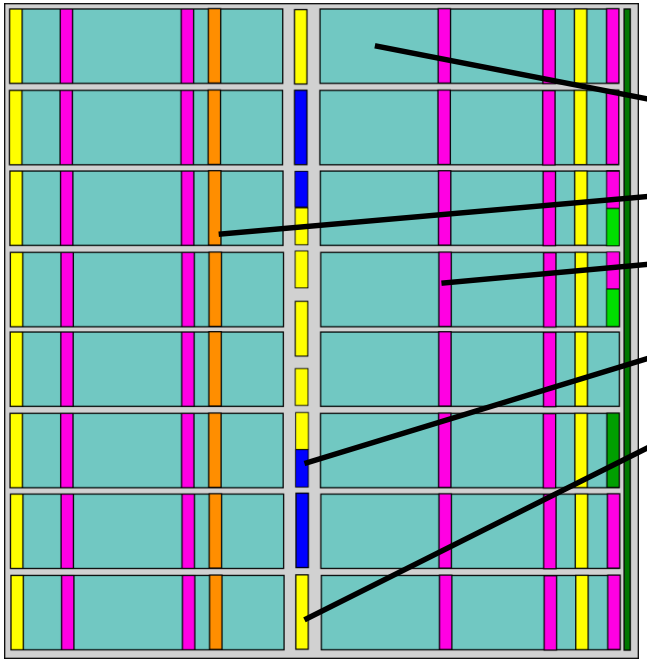
Begriffserklärung µC/DSP

- Watchdog
  - Ein Watchdog (Wachhund) wirkt einem Komplettausfall eines Gerätes durch Softwareversagen entgegen. Die Software muss in regelmäßigen Abständen dem Watchdog mitteilen, dass sie noch richtig funktioniert. Bleibt dies aus, wird ein Reset ausgelöst.

- Zero-Overhead Schleife
  - Wiederholte Ausführung von kleineren Programmteilen ohne zusätzliche Zyklen für das Aktualisieren/Testen des Schleifenzählers und Rücksprung an den Anfang.

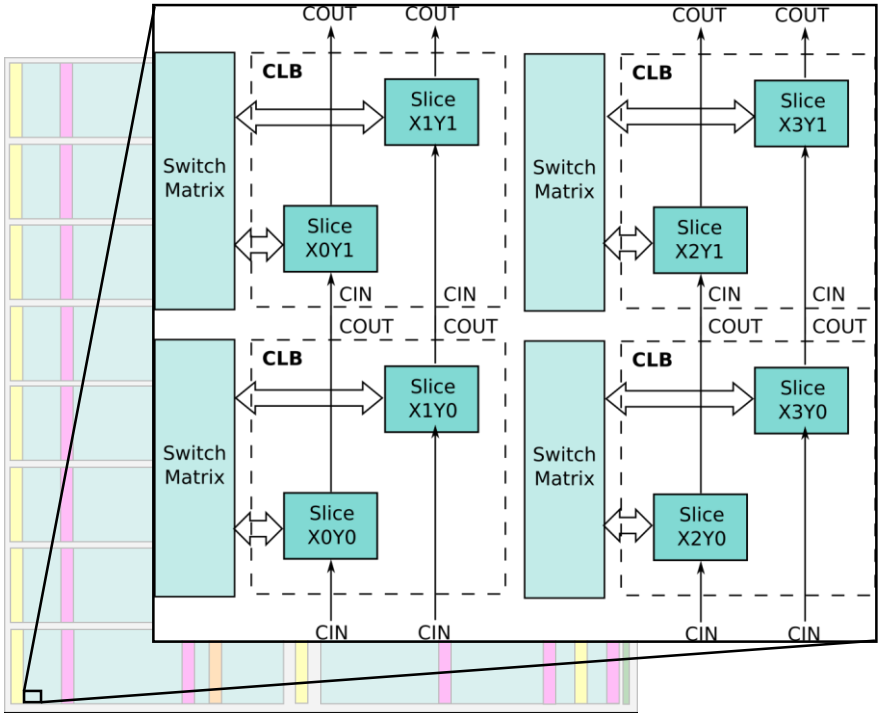
- Multiply Accumulate:
  - Befehlsverkettung von Multiplikation und Addition in einem Befehl, die möglichst in einem Taktschritt ausgeführt werden (`a + b * c`). Mögliche Datentypen sind Integer, Fixed-Point oder Floating-Point. Anwendung für z.B. Kreuzprodukt, Matrixmultiplikation und Polynomberechnung (Horner-Schema).

FPGA: Xilinx Virtex5 110T



- Configurable Logic Block (CLB)
- Digital Signal Processor (DSP)
- Block RAM (BRAM)
- Digital Clock Manager (DCM)
- I/O Bank

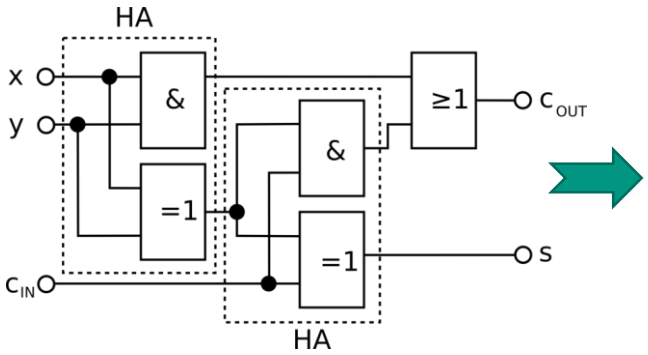
FPGA: Xilinx Virtex5 110T



- One CLB consists of two Slices
- Programmable Switch Matrix
- Fast local routing inside a CLB
- Global routing between CLBs

Volladdierer in einer Slice realisieren

- Wahrheitstabelle aufstellen



X	Y	Cin	S	Cout
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Arbeitsphase

- Aufgabe 2.08: Cache
  - Direct-mapped, 2-Wege Assoziativ und Voll-Assoziativ
  - Lesezugriffe und Verdrängung
  - Cache-Optimierung und Größe
- Aufgabe 2.09: DSP
  - DSP-Optimierungen
- Aufgabe 2.10: Field Programmable Gate Array
  - Volladdierer mittels zwei CLBs

Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

ITIV

Hardware/Software Co-Design

Übung 4 - Lösung

M.Sc. Fabian Lesniak

Lösung Aufgabe 2.08a: Cache

- Wie ist Speicheradresse bei den Verschiedenen Cache-Typen aufgeteilt?
  - Pro Bit der Speicheradresse ist ein Kästchen vorhanden. Jedes Bit kann entweder für den Tag (t), Index (i) oder Byteoffset (o) verwendet werden:

Direct-Mapped	t	t	t	t	i	i	o	o
2-Wege Assoziativ	t	t	t	t	t	i	o	o
Voll-Assoziativ	t	t	t	t	t	t	o	o

Lösung Aufgabe 2.08b: Cache

- 6 Lesezugriffe nacheinander auf den jeweiligen Cache-Typen
- Speicheradressen der Lesezugriffe in Dualform in der ersten Spalte
- In der 2. Spalte (Hit?) wird ein Cache-Hits markiert
- Die restlichen Spalten für den Inhalt der 4 Cache-Zeilen
  - Als Inhalt genügt es, den Tag der Cache-Zeile einzutragen
- Verdrängungsstrategie LRU (Least Recently Used)

Lösung Aufgabe 2.08b: Cache

	Hit?	Line 0	Line 1	Line 2	Line 3
1000 0011	-	1000			
1000 0100	-	1000	1000		
1000 1001	-	1000	1000	1000	
1000 0010	x	1000	1000	1000	
1001 0010	-	1001	1000	1000	
1000 1000	x	1001	1000	1000	

Direct mapped

	Hit?	Line 0	Line 1	Line 2	Line 3
1000 0011	-	10000			
1000 0100	-	10000		10000	
1000 1001	-	10000	10001	10000	
1000 0010	x	10000	10001	10000	
1001 0010	-	10000	10010	10000	
1000 1000	-	10001	10010	10000	

2-Wege assoziativ

	Hit?	Line 0	Line 1	Line 2	Line 3
1000 0011	-	100000			
1000 0100	-	100000	100001		
1000 1001	-	100000	100001	100010	
1000 0010	x	100000	100001	100010	
1001 0010	-	100000	100001	100010	100100
1000 1000	x	100000	100001	100010	100100

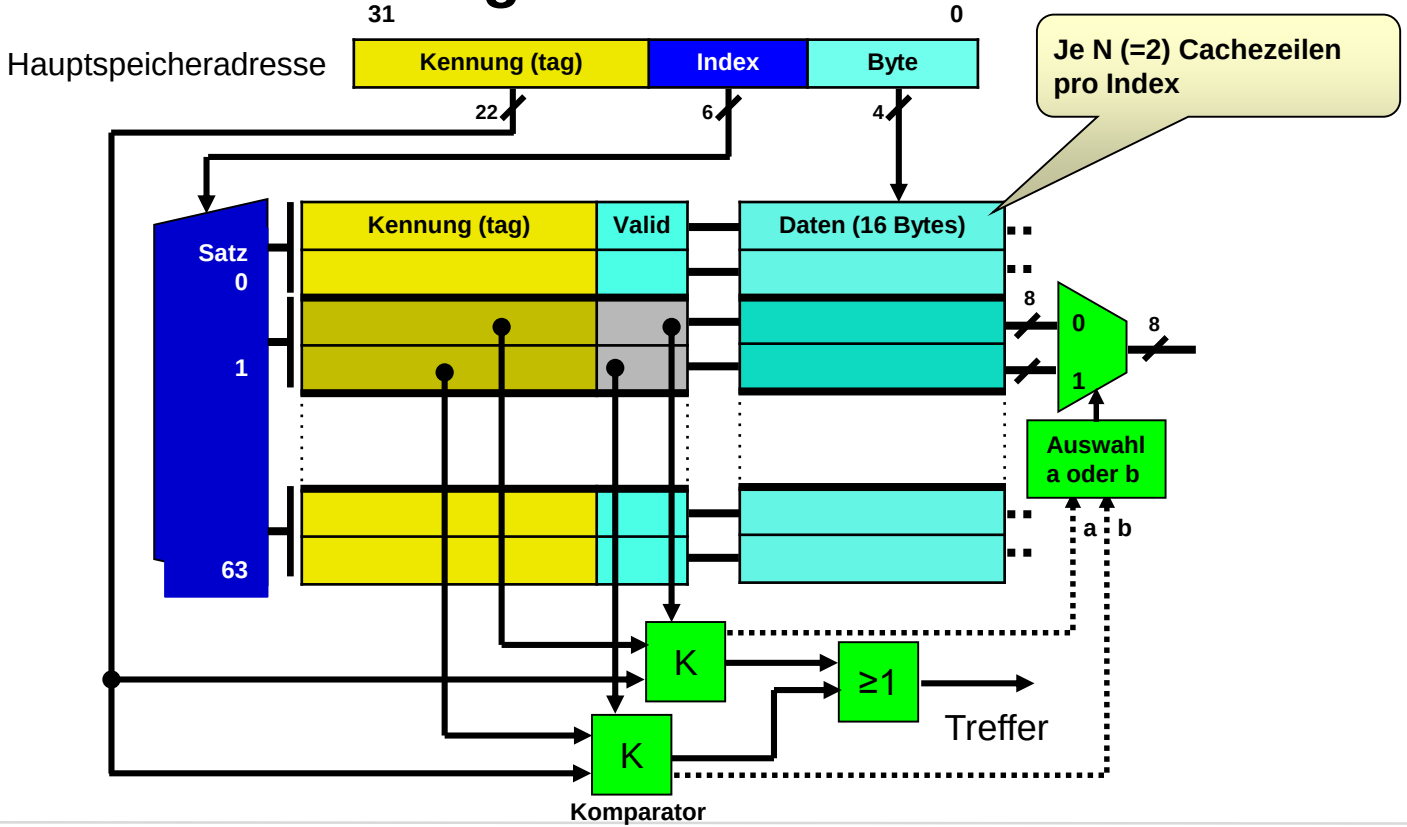
Voll-assoziativ

Lösung Aufgabe 2.08c: Speicherplatz

- Wieviel Speicherplatz verbraucht ein Cache insgesamt? Geben sie ihre Antwort in Abhängigkeit der ihnen bekannten Cache-Parameter an.

  - Es wird Speicherplatz für die **Adresse** und die eigentlichen **Daten** benötigt, die der Cache zwischenspeichern soll. Beides muss jeweils für alle vorhandenen Cache-Lines vorgehalten werden. Von der Adresse muss man **lediglich den Tag** speichern, da Index und Offset direkt für die Ansteuerung der Cache-Line bzw. des Datums in der Cache-Line genutzt werden und daher nicht vorgemerkt werden müssen. Der Speicherplatz für die Daten hängt wiederum von der kleinsten adressierbaren Größe (üblicherweise ein Byte) und der Länge der Cache-Line ab. Zusätzlich kann noch in kleinerem Maße weiterer Speicherplatz z.B. für dirty bits (Cache-Kohärenz) benötigt werden.

2.3.2.5 Caches - 2-Wege assoziativer Cache



Lösung Aufgabe 2.08d: Optimierung

- Eine Matrixmultiplikation wird üblicherweise mit Hilfe von zwei verschachtelten Schleifen implementiert, die über alle Elemente der Matrix laufen (x, y). Wieso gibt es auf einem GPP mit Datencache einen großen Laufzeitunterschied, je nachdem über welche Dimension zuerst (d.h. in der äußeren Schleife) iteriert wird?
  - Eine Matrix wird meist über ein zweidimensionales Array implementiert, welches kontinuierlich im Speicher liegt. Wenn die Matrixmultiplikation in der richtigen Reihenfolge durchgeführt wird, so wird bereits mit der ersten Operation eine Cachezeile geladen, in welcher die Operanden für die nächste Operation mit enthalten sind. Wird sie falsch herum durchgeführt, so tritt bei jeder Operation ein Cache-Miss auf und die Operanden müssen einzeln über den langsamen Hauptspeicher geladen werden.

- Was ist ein Cache?
- Wie sind Offset, Index & Tag aufgeteilt?
- Was ist Assoziativität und wie funktioniert diese?
- Was sind Verdrängungsstrategien?

DSP

- Im Folgenden ist eine Implementierung eines FIR-Filters als C-Code gegeben:

```
sum = 0.0;
for (i=0; i<N; i++)
     sum = sum + a[i]*b[i];
```

- Ein solcher Code kann auf einem DSP unter Ausnutzung der für solche Domänen angepassten Spezialhardware sehr effizient ausgeführt werden. Im Folgenden ist der Code zu sehen, nachdem er optimiert in Assembler übersetzt wurde:

```
RPTS     N -1                  ;Repeat next instruction.
MPYF3    *AR0++%,*AR1++%,R0    ;Multiply…
|| ADDF3 R0,R2,R2              ;… and accumulate.
ADDF     R0,R2                 ;Last product accumulated.
```

DSP

- Wie viele Takte benötigt der optimierte Assembler Code für N = 10?
  - Mit allen DSP Optimierungen: 12 Takte
- Schreiben sie nun das Programm in reinem Assembler ohne alle Spezialbefehle und DSP-Optimierungen. Wie viele zusätzliche Takte braucht das Programm hierdurch für N = 10?

```
loop:  CMPI3    R0, 10
       BZ       9
       LDF      R1, R3
       LDF      R2, R4
       MPYF3    R3, R4, R5
       ADDF3    R5, R6, R6
       ADDI3    R0, 1,  R0
       ADDI3    R1, 4,  R1
       ADDI3    R2, 4,  R2
       BR       loop
```

- Ohne Optimierungen: 102 Takte

- Was ist ein digitaler Signalprozessor?
- Was ist der Unterschied zwischen einem Mikroprozessor und einem digitalen Signalprozessor?
- Wie werden DSPs heute programmiert?
- Wie kann Parallelität realisiert werden?

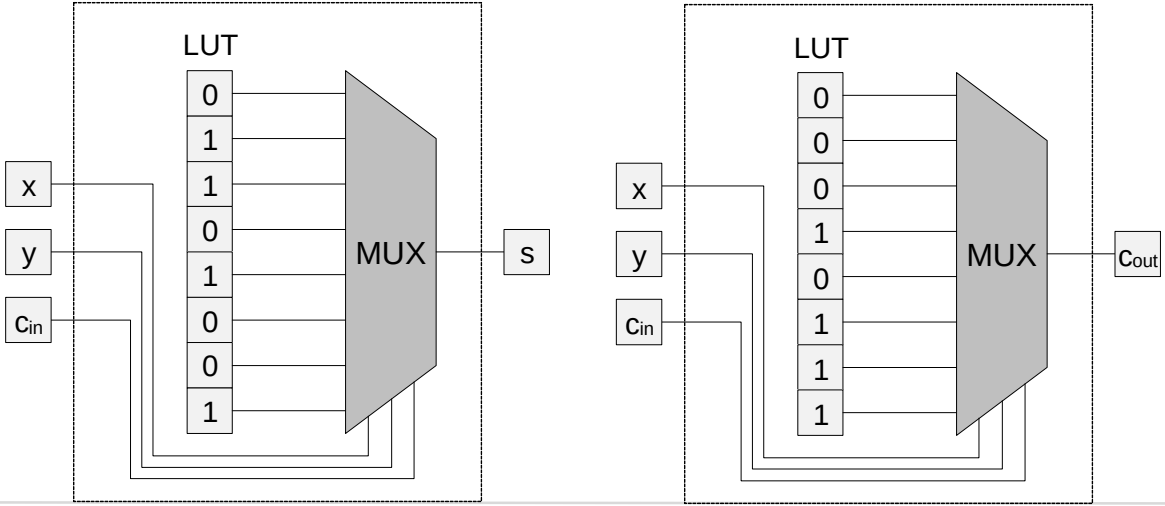
Aufgabe 2.10: FPGA

- Gegeben ist eine einfache Version eines Xilinx FPGAs bestehend aus CLBs und Switch-Matrix. Ein CLB enthalt eine 8 elementige LUT mit 3 Eingängen.
  - Realisieren Sie einen Volladdierer mittels zwei CLBs.

- Ein Volladdierer besitzt drei Eingänge (x, y, cin) und zwei Ausgänge (cout, s). Es werden sämtliche Eingänge addiert und das Ergebnis als 2-Bit Zahl auf die Ausgänge gelegt. So ist s=1 wenn die Summe aller Eingänge ungerade ist und cout=1 wenn die Summe aller Eingänge >= 2 ist.

Lösung Aufgabe 2.10: FPGA

- Gegeben ist eine einfache Version eines Xilinx FPGAs bestehend aus CLBs und Switch-Matrix. Ein CLB enthält eine 8 elementige LUT mit 3 Eingängen.
  - Realisieren Sie einen Volladdierer mittels zwei CLBs.



- Wie ist ein FPGA aufgebaut?
- Wie sieht eine FPGA-Architektur aus?
- Wie wird die Logik bei SRAM-basierten FPGAs abgebildet?
- Wie kann man Schaltungen realisieren?
- Wie wird er programmiert?