Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

**Institutsleitung**
Prof. Dr.-Ing. Dr. h. c. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

# Übung 4

**Dipl.-Inform. Leonard Masing**
**Dr.-Ing. Oliver Sander**

Institut für Technik der Informationsverarbeitung (ITIV)

# Hardware/Software Co-Design

# Agenda

- Wiederholung ausgewählter Themen
  - µC vs. DSP
  - FPGA
- Gruppenarbeit
  - FPGA
  - Design Space, Pareto
  - Exaktheit/Treue
  - Taktschlupf
- Vorstellung der Lösung

# Aufgabe 2.09: µC vs. DSP

- Ordnen Sie die folgenden Eigenschaften einem Mikrokontroller oder einem Digitalen Signalprozessor (DSP) zu.
  - VLIW
  - Multiply Accumulate (MAC)
  - Zero Overhead Schleife
  - Multitasking
  - Zähler und Zeitgeber
  - Sättigungs-Arithmetik
  - WatchDog
  - Floating-Point Arithmetik
  - Homogene Registersätze
  - Kontextwechsel
  - A/D und D/A Wandler
  - Fixed-Point Arithmetik
  - Spezielle Adressierungsarten (z.B. Bit-Reverse)
  - Harvard-Architektur
  - Kontrollflussorientiert
  - Interruptsystem
  - DMA Coprozessor
  - Bit- und Logikoperationen
- Erklären Sie folgende Begriffe:
  - WatchDog
  - Zero-Overhead Schleife
  - Multiply Accumulate

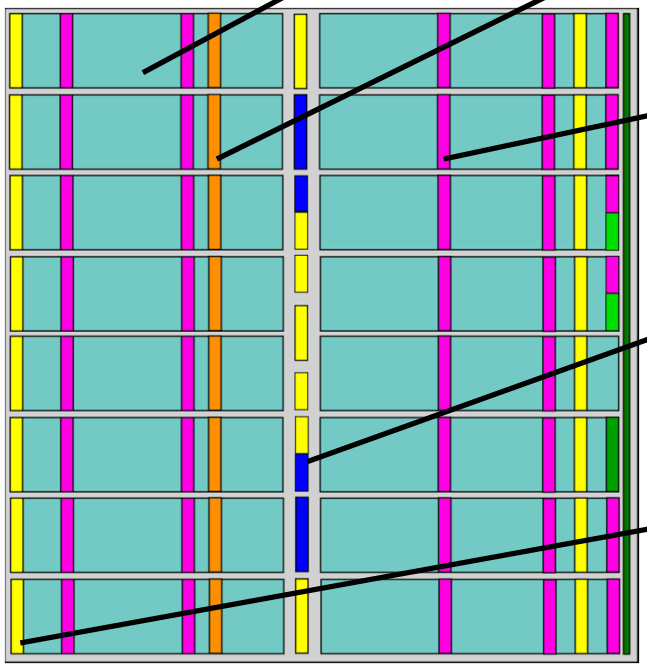
# Begriffserklärung

- **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.

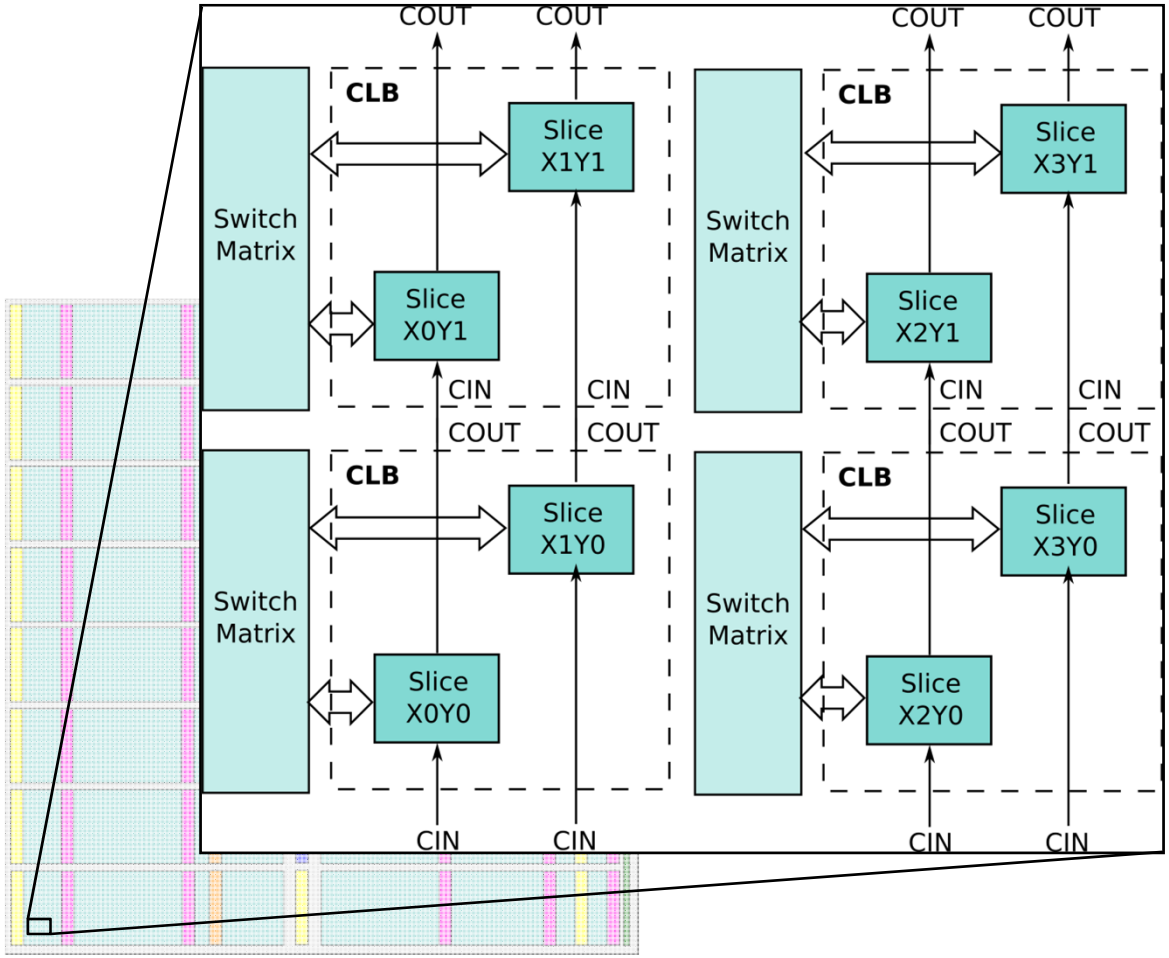
- **Multiply Accumulate:** Befehlsverkettung von Multiplikation und Addition in einem Befehl, die möglichst in einem Taktschritt ausgeführt werden (a = a + b * c). Mögliche Datentypen sind Integer, Fixed-Point oder Floating-Point. Anwendung für z.B. Kreuzprodukt, Matrixmultiplikation und Polynomberechnung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

# FPGA: Xilinx Virtex5 110T

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

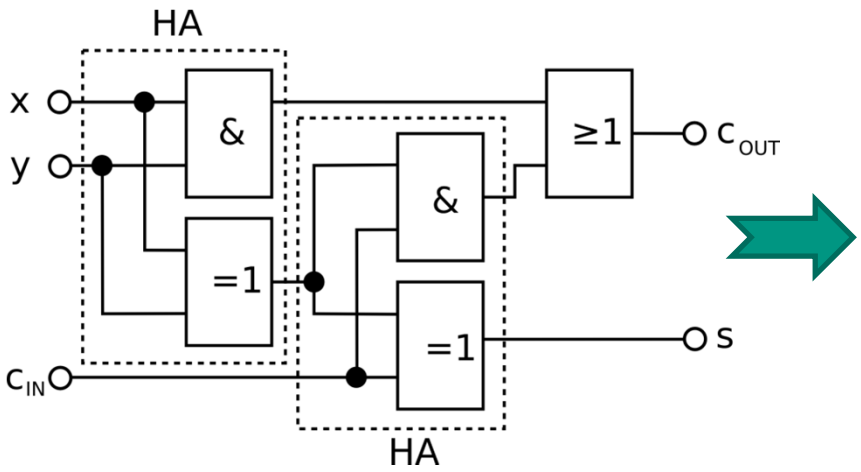
# FPGA : Xilinx Virtex5 110T




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

# How to map a full adder into a Slice

- Create a truth table




| 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 2.10: Field Programmable Gate Array
  - Volladdierer mittels zwei CLBs
- Aufgabe 3.01: Design Space, Pareto Punkte
  - Realisierungsmöglichkeiten mit verfügbaren Komponenten je nach Kosten und Ausführungsgeschwindigkeit
- Aufgabe 3.02: Exaktheit & Treue
  - Metriken und Entwurfsqualität mit geschätzten und gemessenen Werte
- Aufgabe 3.03: Taktschlupf
  - Taktschlupfminimierung, slack, mittlerer Schlupf

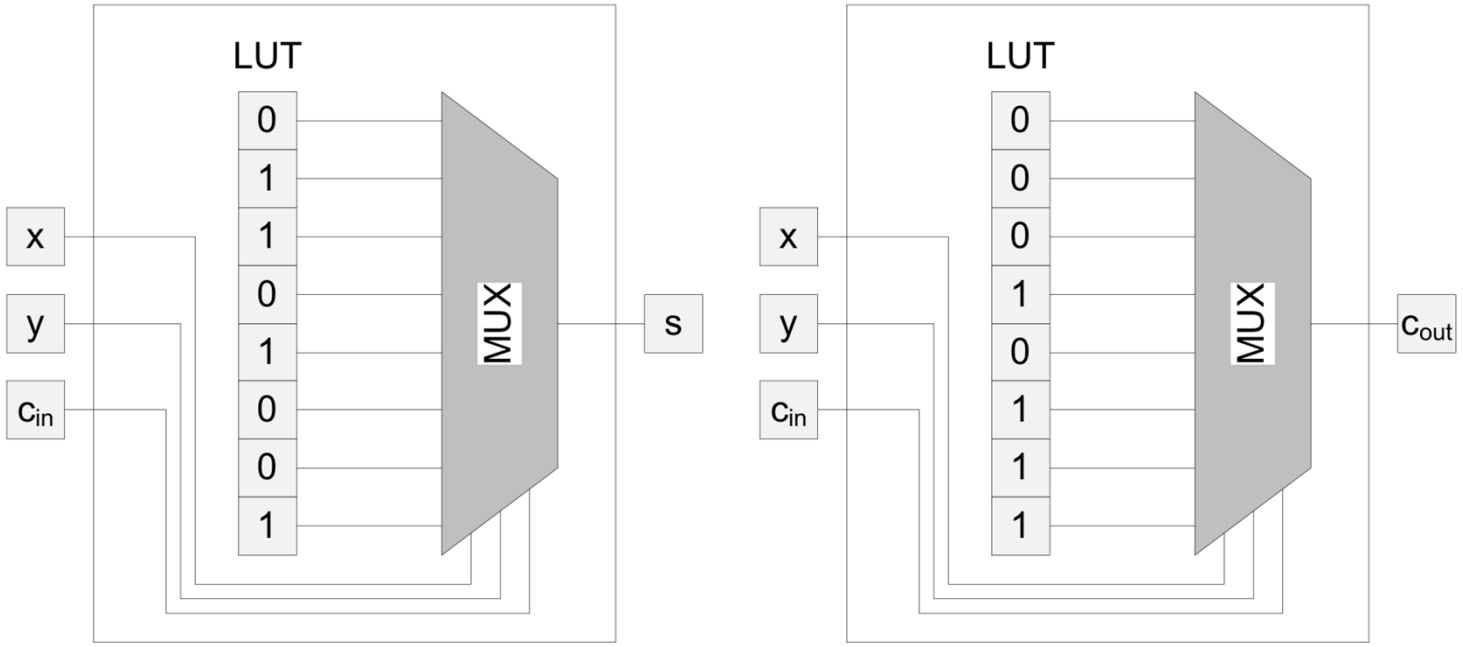
# Aufgabe 2.10: FPGA

- Gegeben ist eine einfache Version eines Xilinx FPGAs bestehend aus CLBs und Switch-Matrixs. Ein CLB enthalt eine 8 elementige LUT mit 3 Eingängen.
  a) 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.

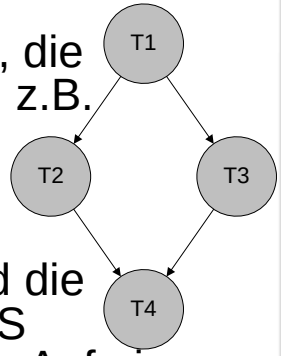
# Lösung Aufgabe 2.10: FPGA

- Gegeben ist eine einfache Version eines Xilinx FPGAs bestehend aus CLBs und Switch-Matrixs. Ein CLB enthält eine 8 elementige LUT mit 3 Eingängen.
  a) 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?

# Aufgabe 3.01: Design Space, Pareto Punkte




- Die rechte Abbildung zeigt einen Task-Graphen mit vier Tasks (T1 … T4), die auf verschiedenen Komponenten ausgeführt werden können. Dabei kann z.B. T4 erst aufgeführt werden, wenn T2 und T3 abgearbeitet wurden. T2 und T3 hingegen können parallel ausgeführt werden.
- Die folgende Tabelle zeigt alle verfügbaren Komponenten (MIPS, DSP, FPGA, ASIC), die maximale Anzahl einer Komponente, deren Kosten und die Ausführungsgeschwindigkeit der vier Tasks. Zum Beispiel kostet der MIPS Prozessor 200 Einheiten und kann Task T1 in 5 und T4 in 2ms ausführen. Auf einer Komponente kann jeweils nur ein Task zur gleichen Zeit ausgeführt werde.

| Komponente | Anzahl | Kosten [€] | Ausführungszeit [ms] | | | |
|---|---|---|---|---|---|---|
| | | | T1 | T2 | T3 | T4 |
| MIPS | 1 | 200 | 5 | - | - | 2 |
| DSP | 1 | 100 | - | 20 | 18 | 5 |
| FPGA | 1 | 250 | - | 12 | 10 | - |
| ASIC | 1 | 400 | - | - | 0,8 | - |

- Vervollständigen Sie die Ausführungszeit und Kosten der folgenden Tabelle. Sie zeigt sämtliche Realisierungsmöglichkeiten, welcher Task auf welchem Prozessor ausgeführt werde kann.
- Tragen Sie die Lösungen aus Aufgabe a) in folgendes Zeit-Kostendiagramm ein und markieren Sie die Pareto-Punkte.
- Was passiert, wenn die Anzahl der Komponenten nicht auf 1 beschränkt ist? Welche zusätzlichen Design-Möglichkeiten ergeben sich? Verändert sich die Menge der Pareto Punkte?

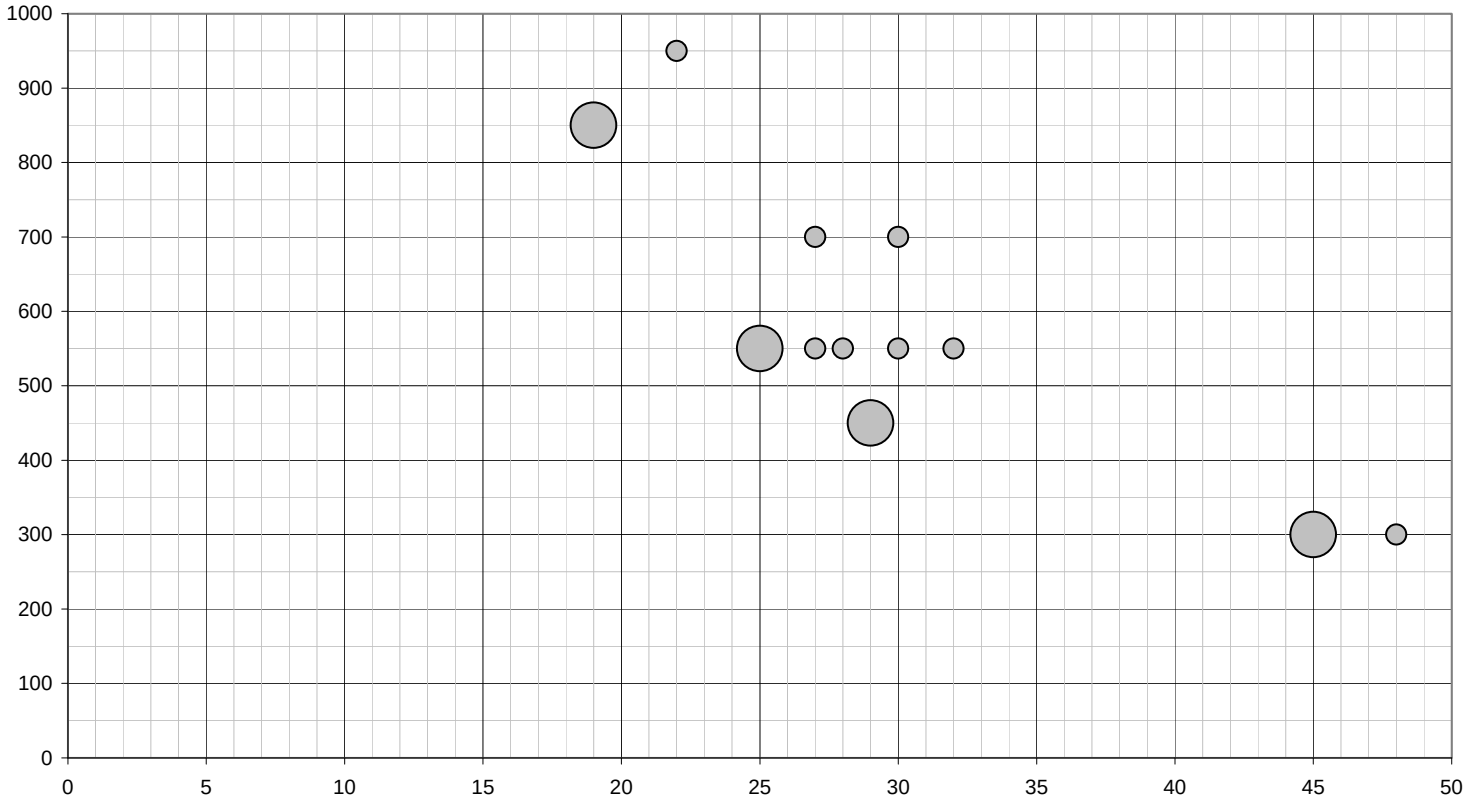
# Aufgabe 3.01: Design Space, Pareto Punkte

- Vervollständigen Sie die Ausführungszeit und Kosten der folgenden Tabelle. Sie zeigt sämtliche Realisierungsmöglichkeiten, welcher Task auf welchem Prozessor ausgeführt werde kann.

| # | Tasks | | | | Ausführungszeit [ms] | Kosten |
|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T4 | | |
| 1 | MIPS | DSP | DSP | MIPS | 5 + 20 + 18 + 2 = 45 | 200+100= 300 |
| 2 | MIPS | DSP | DSP | DSP | 5 + 20 + 18 + 5 = 48 | 200+100= 300 |
| 3 | MIPS | DSP | FPGA | MIPS | 5 + max(20, 18) + 2 = 27 | 200+100+250= 550 |
| 4 | MIPS | DSP | FPGA | DSP | 5 + max(20, 18) + 5 = 30 | 200+100+250= 550 |
| 5 | MIPS | DSP | ASIC | MIPS | 5 + max(20, 0.8) + 2 = 27 | 200+100+400= 700 |
| 6 | MIPS | DSP | ASIC | DSP | 5 + max(20, 0.8) + 5 = 30 | 200+100+400= 700 |
| 7 | MIPS | FPGA | DSP | MIPS | 5 + max(12, 18) + 2 = 25 | 200+250+100= 550 |
| 8 | MIPS | FPGA | DSP | DSP | 5 + max(12, 18) + 5 = 28 | 200+250+100= 550 |
| 9 | MIPS | FPGA | FPGA | MIPS | 5 + 12 + 10 + 2 = 29 | 200+250= 450 |
| 10 | MIPS | FPGA | FPGA | DSP | 5 + 12 + 10 + 5 = 32 | 200+250+100= 550 |
| 11 | MIPS | FPGA | ASIC | MIPS | 5 + max(12, 0.8) + 2 = 19 | 200+250+400= 850 |
| 12 | MIPS | FPGA | ASIC | DSP | 5 + max(12, 0.8) + 5 = 22 | 200+250+400+100=950 |

# Lösung Aufgabe 3.01b: Design Space, Pareto Punkte

- Tragen Sie die Lösungen aus Aufgabe a) in folgendes Zeit-Kostendiagramm ein und markieren Sie die Pareto-Punkte.



# Lösung Aufgabe 3.01c: Design Space, Pareto Punkte

- Was passiert, wenn die Anzahl der Komponenten nicht auf 1 beschränkt ist? Welche zusätzlichen Design-Möglichkeiten ergeben sich? Verändert sich die Menge der Pareto Punkte?

| # | Tasks | | | | Ausführungszeit [ms] | Kosten |
|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T4 | | |
| 13 | MIPS | DSP | DSP | MIPS | 5 + max(20, 18) + 2 = 27 | 200+100+100= 400 |
| 14 | MIPS | DSP | DSP | DSP | 5 + max(20, 18) + 5 = 30 | 200+100+100= 400 |
| 15 | MIPS | FPGA | FPGA | MIPS | 5 + max(12, 10) + 2 = 19 | 200+250+250= 700 |
| 16 | MIPS | FPGA | FPGA | DSP | 5 + max(12, 10) + 5 = 22 | 200+250+250+100=800 |

- Dadurch ändert sich die Menge der Pareto Punkte
  - Lösung #15 ist genauso schnell wie Lösung #11 kostet aber weniger.
  - Lösung #13 überdeckt Lösung #9 in Kosten und Ausführungszeit.

- Was ist Pareto-optimal?
- Welche Graphenmodelle gibt es? Welche Eigenschalften haben sie?
- In welchen Graphen kann Kontrollfluss abgebildet werden?
- Wie wird Programmcode in Graphen dargestellt?

# Aufgabe 3.02: Exaktheit und Treue

- In folgender Tabelle sind für vier Entwurfspunkte Metriken und Entwurfsqualität dargestellt, und zwar geschätzt Werte E(D) sowie die gemessenen Werte M(D).

| Entwurfspunkt | E(D) | M(D) |
|---|---|---|
| W | 112 | 100 |
| X | 128 | 137 |
| Y | 139 | 121 |
| Z | 205 | 132 |

a) Bestimmen Sie die Exaktheit (A) des Entwurfspunktes W.

a) Bestimmen Sie die Treue (F) des Schätzverfahrens

# Lösung Aufgabe 3.02: Exaktheit und Treue

a) Bestimmen Sie die Exaktheit (A) des Entwurfspunktes W.

$$A_W = 1 - \frac{|E(D_W) - M(D_W)|}{|M(D_W)|} = 1 - \frac{|112 - 100|}{|100|} = 1 - \frac{12}{100} = \frac{88}{100} = 0{,}88$$

| Entwurfspunkt | E(D) | M(D) |
|---|---|---|
| W | 112 | 100 |
| X | 128 | 137 |
| Y | 139 | 121 |
| Z | 205 | 132 |

b) Bestimmen Sie die Treue (F) des Schätzverfahrens

$$F = 100\% \cdot \frac{2}{n(n-1)} \cdot \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mu_{i,j}$$

$$= 100\% \cdot \frac{2}{4 \cdot 3} \cdot (\mu_{W,X} + \mu_{W,Y} + \mu_{W,Z} + \mu_{X,Y} + \mu_{X,Z} + \mu_{Y,Z})$$

$$= 100\% \cdot \frac{1}{6} \cdot (1 + 1 + 1 + 0 + 0 + 1) = 100\% \cdot \frac{4}{6} = 67\%$$

# Aufgabe 3.03: Taktschlupf

- Gegeben ist eine Menge von funktionale Einheiten vk. Die mögliche Taktperiode Ihrer Zieltechnologie liegt zwischen 20 und 50 ns.

| Funktionale Einheit | k | delay($v_k$) [ns] | occ($v_k$) |
|---|---|---|---|
| MUL | 1 | 135 | 9 |
| ADD | 2 | 45 | 10 |
| SUB | 3 | 55 | 1 |

a) Was ist der Taktschlupf?
b) Was bringt die Taktschlupfminimierung?
c) Berechnen Sie den „slack“ für alle funktionalen Einheiten bei einer Taktperiode von 20ns.
d) Berechnen Sie den mittleren Schlupf (avgslack) für eine Taktperiode von 20ns.
e) Raten/Überlegen Sie, welche Taktperiode den niedrigsten mittleren Schlupf hat?

# Lösung Aufgabe 3.03: Taktschlupf

a) Was ist der Taktschlupf?

- Der Taktschlupf bezeichnet den Anteil einer Taktperiode, der von einer funktionalen Einheit nicht ausgenutzt wird.

b) Was bringt die Taktschlupfminimierung?

- Bei der Taktschlupfminimierung wird versucht den durchschnittlichen Taktschlupf pro Operation zu minimieren. Dadurch steigt die Taktauslastung der Hardware und somit der Performanz.

# Lösung Aufgabe 3.03: Taktschlupf

c) Berechnen Sie den „slack“ für alle funktionalen Einheiten bei einer Taktperiode von 20ns.

- *slack(20ms,v_MUL )=(⌈delay(v_MUL )/20⌉*20-delay(v_MUL)*
  *=(⌈135/20⌉)*20-135=5*
  *slack(20ms,v_ADD)=(⌈delay(v_ADD )/20⌉*20-delay(v_ADD)*
  *=(⌈45/20⌉)*20-45=15*
  *slack(20ms,v_SUB)=(⌈delay(v_SUB )/20⌉*20-delay(v_SUB)*
  *=(⌈55/20⌉)*20-55=5*

d) Berechnen Sie den mittleren Schlupf (avgslack) für eine Taktperiode von 20ns.Der Taktschlupf bezeichnet den Anteil einer Taktperiode, der von einer funktionalen Einheit nicht ausgenutzt wird.

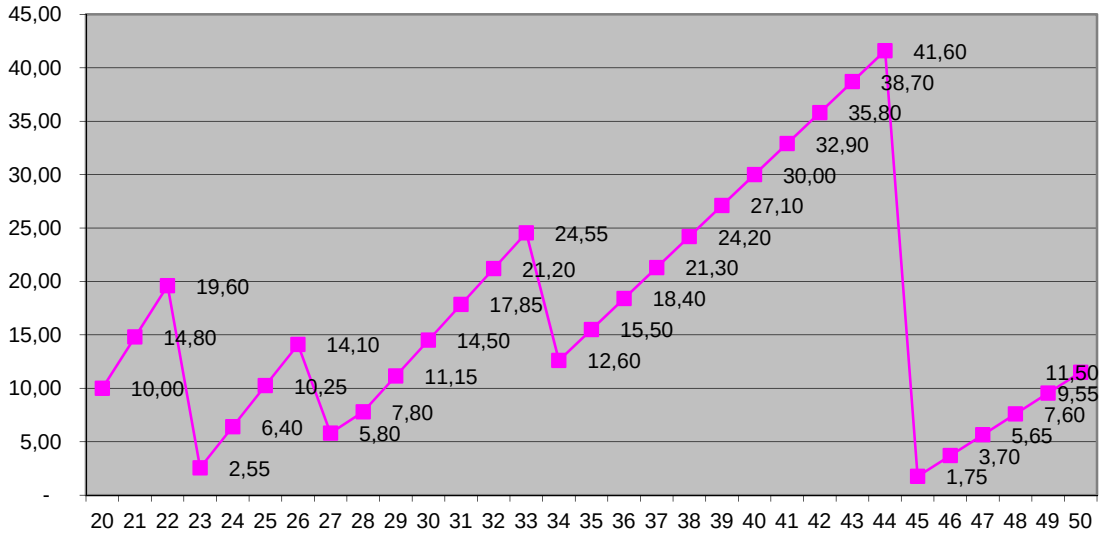
- $$avgslack(T) = \frac{\sum_{k=1}^{|v_T|}(occ(v_k) * slack(20, v_k))}{\sum_{k=1}^{|v_T|} occ(v_k)}$$

  $$avgslack(20ms)$$
  $$= \frac{occ(v_{MUL}) * slack(20, v_{MUL}) + occ(v_{ADD}) * slack(20, v_{ADD}) + occ(v_{SUB}) * slack(20, v_{SUB})}{occ(v_{MUL}) + occ(v_{ADD}) + occ(v_{SUB})}$$
  $$= \frac{9 * 5 + 10 * 15 + 1 * 5}{9 + 10 + 1} = \frac{45 + 150 + 5}{20} = 10$$

# Lösung Aufgabe 3.03: Taktschlupf

e) Raten/Überlegen Sie, welche Taktperiode den niedrigsten mittleren Schlupf hat?

| Funktionale Einheit | k | delay($v_k$) [ns] | occ($v_k$) |
|---|---|---|---|
| MUL | 1 | 135 | 9 |
| ADD | 2 | 45 | 10 |
| SUB | 3 | 55 | 1 |



- Warum erfolgt eine Schätzung der Entwurfsqualität?
- Welche Metriken können geschätzt werden?
- Was ist Hardware-Performanz? Wie setzt sich diese zusammen?
- Wieso wird die Taktperiode geschätzt? Wie kann sie optimiert werden?