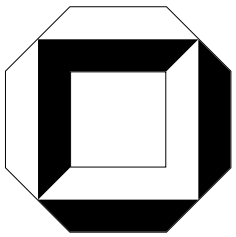


Universität Karlsruhe (TH)
Institut für Technische Informatik
Prof. Dr. Wolfgang Karl

Klausur Rechnerstrukturen
Sommersemester 2007
Musterlösung

Aushang der Ergebnisse: ab Ende September 2007

Musterlösung 1: Quantifizierung *10P*

a) SPECint ($\frac{1}{2}$P): Es handelt sich hierbei um eine per Benchmark ($\frac{1}{2}$P) (genauer: SPEC ist eine Sammlung von Benchmark-Programmen) ermittelte Leistungsquantifizierung, die durch den Bezug auf eine Referenzmaschine einen fairen Vergleich zwischen Systemen erlaubt ($\frac{1}{2}$P). **1,5P**

b) $MIPS = \frac{f}{CPI*10^6}$ **0,5P**

c) Zugriffszeit t_z und Übertragungszeit $t_{ü}$ *($\frac{1}{2}$P für vollständige Bezeichnungen).* **1,5P**

Hieraus leiten sich Bedienzeit X_i und maximale Auslastung D_{imax} wie folgt ab:

($\frac{1}{2}$P) $X_i = t_z + t_{ü}$

($\frac{1}{2}$P) $D_{imax} = \frac{1}{X_i} = \frac{1}{t_z + t_{ü}}$

d) Prozessor 2($\frac{1}{2}$P): Der niedrigere MIPS-Wert resultiert in kompakterem Code (weniger Speicherverbrauch, $\frac{1}{2}$P) sowie niedrigere Schaltfrequenzen und damit niedrigeren Energieverbrauch ($\frac{1}{2}$P). **1,5P**

e) ($\frac{1}{2}$P) $dpw = \frac{\pi*(d_{wafer}*\frac{1}{2})^2}{a_{die}} - \frac{\pi*d_{wafer}}{\sqrt{2*a_{die}}} = A - B$ **2P**
($\frac{1}{2}$P) A: Gesamtfläche, B: Verschnitt
(1P) Der Anteil der Gesamtfläche wächst quadratisch zum Wafer-Durchmesser, der Anteil des Verschnitts nur linear.

f) ($\frac{1}{2}$P) $yield_{die} = yield_{wafer} * (1 + \frac{dpua*a_{die}}{\alpha})^{-\alpha} \leftrightarrow \frac{yield_{wafer}}{yield_{die}} = 1 + dpua * a_{die}$ **1P**

($\frac{1}{2}$P) $1 = 1 + dpua * a_{die} \leftrightarrow dpua * a_{die} = 0 \rightarrow dpua = 0$

g) ($\frac{1}{2}$P) Die-Kosten: $cost_{die} = \frac{cost_{wafer}}{dpw*yield_{die}}$ **1P**

($\frac{1}{2}$P) IC-Kosten: $cost_{ic} = \frac{cost_{die}+cost_{test}+cost_{packaging}}{yield_{final}}$

h) Offenkundig Test- und Packaging-Kosten ($\frac{1}{2}$P), denn andere Fertigungsfehler sind bereits im Die-Yield enthalten ($\frac{1}{2}$P). **1P**

Musterlösung 2: Hardwareentwurf *10P*

a) *(je $\frac{1}{2}P$ für Verfeinerung und Erklärung)* **3P**

- Verhaltensverfeinerung: Ersetzen von Black-Boxes (d.h. detailliertes Ausarbeiten der gewünschten Funktionalität)
- Strukturverfeinerung: Verschaltung von Kompenenten mit einfacher Funktionalität zur Realisierung einer spezifizierten Funktion
- Datenverfeinerung: Realisierung abstrakter Datentypen durch einfachere Datentypen (typischerweise `bit` oder `std_logic`)

b) *(je $\frac{1}{2}P$ für Teil und Funktion)* **2P**

- entity: Schnittstellendefinition
- architecture: Verhaltensbeschreibung

Hinweis: `configuration` *ist nicht zwingend.*

c) **3P**

- `count` ist undefiniert ($\frac{1}{2}$P), der Wert `count+1` („undefiniert+1“) ist daher ebenfalls undefiniert ($\frac{1}{2}$P). *1P*
- Initialisierung (z.B. mittels Rücksetzsignal) *0,5P*
- `process(clk,count)` → `process(clk,count,rst)` *($\frac{1}{2}P$)* *1,5P*

```
if clk'event and clk='1' then →
   if rst='1' then count<=''00000000''
   elsif clk'event...
```

($\frac{1}{2}P$ für Konstrukterweiterung um Reset-Zweig, $\frac{1}{2}P$ für Korrektheit)

d) **2P**

- Die Zählerinkrementierung wird erst mit einer Verzögerung von einem Taktzyklus sichtbar, darum wird das Signal `flag` tatsächlich den Zählerstand `0x00` signalisieren. *1P*
- `flag<='1' when count=X"ff" else '0';` *1P*
  ($\frac{1}{2}P$ für Konstrukt, $\frac{1}{2}P$ für Korrektheit)

Musterlösung 3: Prozessorarchitektur *10P*

a) **1P**

- $T = n + k - 1 = 800000 - 7 + 1 = 800006$ Zyklen *0,5P*
- $s = \frac{Pipelinetiefe}{1+stall-Zyklen} = \frac{7}{1+2,5} = \frac{7}{3,5} = 2$ *0,5P*

b) Die langsamste Stufe bestimmt den Takt ($\frac{1}{2}$P), also $f = \frac{1}{10ns} = 100MHz$ ($\frac{1}{2}$P). **1P**

c) **2P**

- Echte Datenabhängigkeit ($\frac{1}{2}$P), führt zu Datenkonflikt und erzwingt somit Anhalten der Pipeline (stall) für zwei Taktzyklen ($\frac{1}{2}$P) *1P*
- Einfügen von Leerzyklen mittels NOP (oder anderen Befehlen) *0,5P*
- Result-Forwarding (und zwar vom Ausgang zum Eingang der EX-Stufe) *0,5P*

d) **1,5P**

- X[5*i+3] → a=5, b=3
- 7*X[3*i+5]+8 → c=3, d=5
- Test: (d-b) mod GCD(c,a)=0 *0,5P*
- $5 - 3 = 2;\ GCD(3,5) = 1;\ 2\ mod\ 1 = 0$, d.h. Abhängigkeit ist potentiell möglich. *0,5P*
- Das GCD-Verfahren trifft nur für den negativen Testfall eine definitive Aussage (d.h. es kann nur mit Sicherheit festgestellt werden, dass keine Abhängigkeiten bestehen). *0,5P*

e) *je Fehler $\frac{1}{2}$P Abzug* **4P**

Ein-	Sprung 1				Sprung 2			
sprung	Prädiktor	Vorhersage	Sprung	P. neu	Prädiktor	Vorhersage	Sprung	P. neu
T	(T, **NT**)	NT	NT	(T,NT)	(**NT**, T)	NT	T	(T, T)
–	(T, **NT**)	NT	T	(T, T)	(T,**T**)	T	NT	(T,NT)
–	(**T**, T)	T	T	(T, T)	(T,**NT**)	NT	T	(T, T)
–	(T, **T**)	T	NT	(T,NT)	(**T**, T)	T	NT	(NT, T)

f) Branch-History-Register (BHR) durch globale Sprunghistorie **0,5P**

Musterlösung 4: Parallelverarbeitung *10P*

a) $S(n) = \frac{T(1)}{T(n)}$, d.h.$S(128) = \frac{2560}{40} = 64$ ($\frac{1}{2}$P) **1P**
$E(n) = \frac{S(n)}{n}$, d.h. $E(128) = \frac{S(128)}{128} = \frac{64}{128} = 0.5$ ($\frac{1}{2}$P)

b) $R(n) = \frac{P(n)}{P(1)}$, d.h. $R(128) = \frac{2{,}56*10^{12}}{2*10^{12}} = 1,28$($\frac{1}{2}$P) **1P**
$U(n) = R(n) * E(n)$, d.h. $U(128) = R(128) * E(128) = 1,28 * 0,5 = 0,64$ ($\frac{1}{2}$P)

c) Amdahls Gesetz: $T(n) = T(1) * (\frac{(1-a)}{n} + a)$ **1P**
$\rightarrow 40 = 2560 * (\frac{1-a}{128} + a) = 2560 * \frac{1+127a}{128} = 20 * (1 + 127a)$
$\rightarrow 20 = 2540a \rightarrow a = \frac{20}{2540} = \frac{1}{127}$

d) $T(min) = \lim_{n\to\infty}(T(1) * (\frac{(1-a)}{n} + a)) = \lim_{n\to\infty}(T(1) * \frac{(1-a)}{n} + T(1) * a)$ **1P**
$T(min) = 2560 * \frac{1}{128} = 20$ ($\frac{1}{2}$P)
$S(max) = \frac{T(1)}{T(min)=}\frac{2560}{20} = 128$ ($\frac{1}{2}$P)

e) Dekomposition: Zerlegung der sequentiellen Anwendung in einzelne Tasks ($\frac{1}{2}$P) **2P**
Zuweisung: Zuweisung der einzelnen Tasks zu Prozessen ($\frac{1}{2}$P)
Festlegung: Festlegung der Kommunikation der Prozesse ($\frac{1}{2}$P)
Abbildung: Zuordnung der Prozesse auf Prozessoren ($\frac{1}{2}$P)

f) SIMD-Verarbeitung ($\frac{1}{2}$P), d.h Verarbeitung von Vektoren in einem Rechenwerk mit pipelineartig aufgebauten Funktionseinheiten ($\frac{1}{2}$P), unter Zuhilfenahme von Vektoroperationen ($\frac{1}{2}$P). **1,5P**

g) **1,5P**

- Knotenanzahl: $4^5 = 16 * 16 * 4 = 1024$ *0,5P*
- Verbindungsgrad: $2 * (n - 1) = 10$ *0,5P*
- Durchmesser: $n * \lfloor \frac{k}{2} \rfloor = 10$ *0,5P*

h) *1P*

- Diskonnektivität = # Knoten / min. Bisektionsbreite *0,5P*
- Kosteneffektivität = Verbindungsgrad * max(Durchmesser, Diskonnektivität) *0,5P*

Musterlösung 5: Speicherhierarchie *10P*

a) *(je Fehler $\frac{1}{2}$P Abzug)* **1,5P**
Berechung: #Erfolgreiche Speicherzugriffe pro Hierarchiestufe = # Speicherzugriffe * Hitrate
$L_1 : 1000 * 0,7 = 700$
$L_2 : 300 * 0,5 = 150$
$L_3 : 150 * 0,6 = 90$
$HS : 60 * 1 = 60$

b) (*Ansatz 1P, Ergebnis $\frac{1}{2}$P)* **1,5P**
$t_a = t_{L1} * r_{H1} + (1 - r_{H1}) * (t_{L2} * r_{H2} + (1 - r_{H2}) * (t_{L3} * r_{H3} + t_{HS} * (1 - r_{H3})))$
$t_a = 1ns*70\%+(1-70\%)*(5ns*50\%+(1-50\%)*(15ns*60\%+150ns*(1-60\%)))$
$t_a = 11,8ns$
Alternativ (vgl. Aufgabe 5a):
$t_a = \frac{\sum_{\#Speicherhierarchien}(\#Zugriffe*Zugriffszeit)}{\#Speicherzugriffe} = \frac{700*1+150*5+90*15+60*150}{1000} = 11,8$

c) (*je Fehler $\frac{1}{2}$P Abzug*) **2,5P**

Adresse	Set-Reuse Distance	Satz
0x0000	∞	0
0x0004	∞	1
0x0100	∞	0
0x0008	∞	0
0x0000	2	0
0x0100	2	0
0x0008	2	0
0x000C	∞	1
0x0104	∞	1
0x0004	2	1
0x0100	1	0
0x0000	2	0

d) Hitrate = 1 / 12 **0,5**

e) *($\frac{1}{2}$P Abzug pro Fehler)* **4P**

Prozessor	Aktion	Prozessor 1			Prozessor 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
	init	1/E	2/S	4/E	2/S	6/M	-
1	rd 6	6/S				6/O	
2	rd 2		2/S		2/S		
1	wr 6	6/M				6/I	
1	wr 4			4/M			
2	rd 3					3/E	
1	rd 5		5/E				
2	rd 6	6/O					6/S
2	wr 5		5/I		5/M		

Musterlösung 6: Fehlertoleranz *10P*

a) **1P**

- Ein System arbeitet zuverlässig, wenn es – im Rahmen der Spezifikation – die korrekte, d.h. spezifizierte Funktion erbringt. *0,5P*
- Die Fehlertoleranzeigenschaft ermöglicht es dem System, auch mit einer begrenzten Anzahl fehlerhafter Subsysteme diese spezifizierte Funktion zu erbringen. *0,5P*

b) Es handelt sich um die sogenannte Badewannenkurve ($\frac{1}{2}$P). Deren drei Phasen sind Frühausfall-, Betriebs- und Altersausfallphase ($\frac{1}{2}$P). **1P**

c) **1,5P**

- Hot Standby: Ersatzkomponente befindet sich in ständiger Bereitschaft *0,5P*
- Cold Standby: Ersatzkomponente muss erst aktiviert/hochgefahren werden. *0,5P*

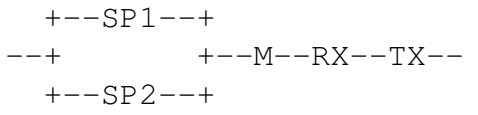
Es handelt sich in beiden Fällen um ungenutzte Redundanz ($\frac{1}{2}$P).

d) Graceful Degradation. Hierbei handelt es sich um gegenseitige Redundanz. **1P**

e) Es findet ein Triple-Modular-Redundancy-System (TMR) mit Entscheider (Voter) Verwendung ($\frac{1}{2}$P). Hierbei handelt es sich um einen Spezialfall der sogenannten n-aus-m-Redundanzsysteme ($\frac{1}{2}$P). Die Funktionswahrscheinlichkeit für diese beträgt **1,5P**

$$\Phi(\tbinom{n}{m}) = \Phi(V) * \sum_{k=n}^{m} \tbinom{m}{k} * \Phi(K)^k * (1 - \Phi(M))^{(m-k)} \quad (\frac{1}{2}P)$$

f) *(je Fehler $\frac{1}{2}P$ Abzug)* **1P**



(RX/TX-Reihenfolge ist für die Funktionsweise unerheblich.)

g) *(je Fehler $\frac{1}{2}P$ Abzug)* **1P**
$S = (SP1 \vee SP2) \wedge M \wedge RX \wedge TX$

h) *($\frac{1}{2}P$ Abzug für falsche SP-Berechnung, sofern restliche Formel korrekt)* **1P**
SP parallel, d.h. $\Phi(SP_p) = 1 - (1 - \Phi(SP))^2$
Somit gilt für das Gesamtsystem: $\Phi(S) = \Phi(TRX)^2 * \Phi(M) * (1 - (1 - \Phi(SP))^2)$

i) Ausfall von Sende- oder Empfangseinheit führt zum Ausfall der Sonde ($\frac{1}{2}$P), darum Verdopplung der jeweiligen Einheiten ($\frac{1}{2}$P). **1P**