Karlsruher Institut für Technologie
Institut für Technische Informatik
Prof. Dr. Wolfgang Karl

Klausur Rechnerstrukturen
Wintersemester 2010/11
Musterlösung

Aushang der Ergebnisse: ab Mitte April 2011

Musterlösung 1: Parallelverarbeitung **11P**

Quantitative Maßzahlen **3P**

a) Amdahls Gesetz (Je 0,5P für Formel und Erklärung): *1P*

$$T(n) = \underbrace{\frac{T(1)}{n} * (1-a)}_{1} \; + \; \underbrace{T(1) * a}_{2}$$

Die Formel zerfällt in die Ausführungszeit des parallel ausführbaren Programmteils 1 und den rein sequentiell ausführbaren Programmteil 2. Es gilt: a mit $(0 \leq a \leq 1)$ ist der der Anteil des Programms, der nur sequentiell ausgeführt werden kann.

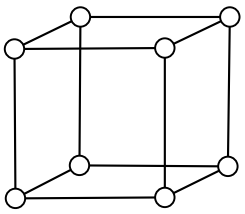
b)
- (0,5P) Das Verhalten heißt superlineare Beschleunigung (superlinearer Speed-Up) *1P*
- (0,5P) Es widerspricht der Abschätzung $1 \leq S(n) \leq n$

c) Die Auslastung ist eine obere Schranke für die Effizienz: *1P*

$$\frac{1}{n} \leq E(n) \leq U(n) \leq 1$$

Verbindungsstrukturen **3P**

d) Radius $K = 2$, Dimension $n = 4$ *1P*



e)
- (0,5P) $N = 2^n$ hier also $N = 2^4 = 16$ *1P*
- (0,5P) Jede Erweiterung benötigt mindestens die Verdoppelung der Knotenzahl.

f)
- (0,5P) Bei einem Verbindungsnetzwerk mit vollständiger Verbindung ist jeder Knoten mit jedem anderen Knoten verbunden. *1P*
- (0,5P) Nachteil: Schlechte Erweiterbarkeit (Netzwerkkosten steigen quadratisch mit der Anzahl der Knoten)

Parallele Architekturen und Parallelverarbeitung: **3P**

g) (pro falscher Antwort -$\frac{1}{2}$P) *1P*
- NORMA: No Remote Memory Access
- NUMA: Non-Uniform Memory Access
- UMA: Uniform Memory Access

h) (pro falscher Antwort -$\frac{1}{2}$P) *1P*

Multiprozessor mit gemeinsamem Speicher	UMA
Multiprozessor mit verteiltem Speicher	NORMA
Multiprozessor mit verteiltem gemeinsamen Speicher	NUMA

i) *1P*
- (0,5P) Die Programmierung erfolgt mit Hilfe des Nachrichtenorientierten Programmiermodells (Message Passing)
- Kommunikation der Prozesse (Threads) mit Hilfe von Nachrichten, kein gemeinsamer Adressbereich
- Verwendung von korrespondierenden Send- und Receive-Operationen

Vektorverarbeitung: **2P**

j) *1P*
- (0,5P) Vorteil: Hardware und Steuerung sind relativ einfach.
- (0,5P) Nachteil: Mehrere unabhängige Pipelines werden benötigt, um alle wichtigen Verknüpfungen durchführen zu können.

k) Die Elemente eines Vektors liegen nicht in aufeinander folgenden Speicherzellen. Ein Vektor Stride gibt den Abstand zwischen Elementen an und ermöglicht den Zugriff auf nicht sequentielle Speicherzellen. *1P*

Musterlösung 2: Fragen des Rechnerentwurfs *10P*

Low-Power-Entwurf **4P**

a) *2,5P*

$$P_{\text{Gesamt}} = P_{\text{Switching}} + P_{\text{Static}} + P_{\text{Leakage}} + P_{\text{Shortcircuit}}$$

(0,5P)

$P_{\text{Switching}}$: Leistungsaufnahme beim Schalten oder Laden einer kapazitiven Last (0,5P)

P_{Static}: Statische Leistungsaufnahme der Schaltung; inhärent in der Schaltung vorhanden (0,5P)

P_{Leakage}: Leckstrom wegen endlich großer Widerstände zwischen Leiterbahnen (0,5P)

$P_{\text{Shortcircuit}}$: Leistungsaufnahme aufgrund von Kurzschluss im CMOS-Gatter bei Änderung der Eingänge (0,5P)

b) $\mathbb{P}_{\text{Schalt}} = 2 * \mathbb{P}_{\text{Ausgangssignal}}(1) * (1 - \mathbb{P}_{\text{Ausgangssignal}}(1))$ (0,5P) *1,5P*

$\mathbb{P}_{\text{Ausgangssignal XOR}}(1) = \mathbb{P}_{\text{Eingang 1}}(0) * \mathbb{P}_{\text{Eingang 2}}(1) + \mathbb{P}_{\text{Eingang 1}}(1) * \mathbb{P}_{\text{Eingang 2}}(0) = \frac{1}{2} * \frac{3}{4} + \frac{1}{2} * \frac{1}{4} = \frac{3}{8} + \frac{1}{8} = \frac{1}{2}$ (0,5P)

$\mathbb{P}_{\text{Schalt}} = 2 * \frac{1}{2} * (1 - \frac{1}{2}) = 2 * \frac{1}{4} = \frac{1}{2}$ (0,5P)

Schaltungsentwurf **6P**

c) Prinzipien zur Implementierung einer Architecture *2P*

- Strukturell (0,5P): Aufbau eines größeren Ganzen mittels Instanziierung und Verdrahtung von Komponenten (0,5P)
- Verhalten (0,5P): Beschreibung des Verhaltens der Schaltung mittels Prozessen, häufig synchron. (0,5P)
- Gemischt: Instanziierung von Komponenten und zusätzliche Verhaltensbeschreibung in Prozessen; häufig in Testbenches anzutreffen

d) *2P*

- Signal wird in Architektur definiert (0,5P) und am Ende des größten umgebenden Blocks gültig gemacht, etwa am Ende eines taktsynchronen Prozesses (0,5P)
- Variablen werden in Prozessen definiert (0,5P) und sofort gültig gemacht (0,5P)

e) Die Beschreibung implementiert einen einfachen Zähler bis 128. (0,5P) Nach 64 Takten wird das Signal flag auf 1 gesetzt. (0,5P) *2P*

Die darauffolgenden 127 Takte ist flag auf 0 gesetzt, dann wieder für die Dauer eines Takts auf 1 usw. (0,5P) da die Abfrage auf 63 nicht den Überlauf des Zählers kennzeichnet, sondern dieser erst nach 127 geschieht (0,5P).

Musterlösung 3: Speicherhierarchie **10P**

a) $t_a = r_{H-L1} * t_{H-L1} + (1 - r_{H-L1}) * (r_{H-L2} * t_{H-L2} + ((1 - r_{H-L2}) * t_{Mem}))$ **1P**

b) A: **2P**

$t_A = 0.8 * 2.5\,ns + 0.2 * (0.9 * 12.5\,ns + 0.1 * 112.5\,ns)$
$t_A = 2\,ns + 0.2 * (11.25\,ns + 11.25\,ns) = 2\,ns + 0.2 * 22.5\,ns$
$t_A = 2\,ns + 4.5\,ns = 6.5\,ns$

B:

$t_B = 0.7 * 2\,ns + 0.3 * (0.9 * 10\,ns + 0.1 * 100\,ns)$
$t_B = 1.4\,ns + 0.3 * (9\,ns + 10\,ns) = 1.4\,ns + 0.3 * 19\,ns$
$t_B = 1.4\,ns + 5.7\,ns = 7.1\,ns$

$t_A < t_B$, Entwurfsalternative A ist schneller und daher zu wählen.

c) Bus-Snooping($\frac{1}{2}$P), welches einen gemeinsamen Datenbus voraussetzt ($\frac{1}{2}$P). **1P**

d) Je richtiger Antwort $\frac{1}{2}$P **2P**

- 2 Zustandsbit in jeder Cachezeile zur Speicherungs des MESI-Zustandes
- Bus-Monitor zur Überwachung des Speicherbusses
- Retry, Abort und Invalidate Signale zu weiteren Prozessoren
- Erweiterung des Cache-Kontrollers um den MESI-Zustandsautomat

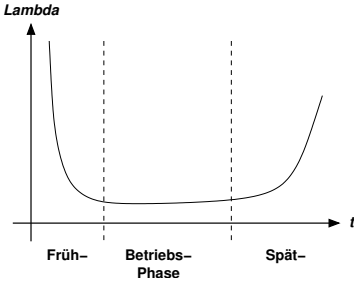
e) *($\frac{1}{2}$P Abzug pro Fehler)* **4P**

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

Musterlösung 4: Fehlertoleranz **10P**

Ausfallverhalten **2P**

a) Badewannenkurve ($\frac{1}{2}$P) *2P*



- Frühphase - Initiale Ausfälle und Fertigungsfehler ($\frac{1}{2}$P)
- Betriebsphase - konstante Ausfallrate ($\frac{1}{2}$P)
- Spätphase - altersbedingte Ausfälle ($\frac{1}{2}$P)

RAID 0-System **4P**

b) Zuverlässigkeitsblockdiagramm: *1P*

```
--FP1--FP2--K--N--KB1--KB2--
```

c) Systemfunktion: $S = FP_1 \wedge FP_2 \wedge K \wedge N \wedge KB_1 \wedge KB_2$ *1P*

d) Ausfälle der einzelnen Komponenten unabhängig voneinander $\Rightarrow$ Ausfallrate des Gesamtsystems λ ist gleich Summe der Ausfallraten der einzelnen Komponenten. *2P*
Es gilt: $\lambda = 1/MTTF$
$\lambda = \frac{2}{100k} + \frac{1}{100k} + \frac{1}{100k} + \frac{2}{200k}$
$= \frac{4}{200k} + \frac{2}{200k} + \frac{2}{200k} + \frac{2}{200k}$
$= \frac{10}{200k} = \frac{1}{20k}$
$MTTF = 1/\lambda = 20k\, Stunden$

RAID 5-System **4P**

e) Funktionswahrscheinlichkeit für ein n-aus-m System: *1P*
$\varphi_m^n = \sum_{k=n}^{m} \binom{m}{k} * \varphi(K)^k * (1 - \varphi(K))^{(m-k)}$

f) Gegenseitige Redundanz ($\frac{1}{2}$P), ermöglicht abgestuften Leistungsabfall bzw. Graceful Degredation ($\frac{1}{2}$P). *1P*

g) Zuverlässigkeitsblockdiagramm *2P*

```
  +--FP1--KB1--FP2--KB2--+
  +                      +
--+--FP1--KB1--FP3--KB3--+--K--N--
  +                      +
  +--FP2--KB2--FP3--KB3--+
```

Musterlösung 5: Quantifizierung **10P**

Analytische/empirische Leistungsbewertung:: 3P

a) Gesetz von Little & Rechnung: $Q = W * D \Leftrightarrow W = \frac{Q}{D}$, für $D \neq 0$, Q: Anzahl von Aufträgen in der Warteschlange, W: Wartezeit, D: Durchsatz *1P*
Setze $Q_A = 22$: $\Rightarrow D_A = 120\frac{\text{Jobs}}{\text{Stunde}} = 2\frac{\text{Jobs}}{\text{Minute}}$
$\rightarrow W = \frac{Q}{D}$, eingesetzt $W_A = \frac{22}{2}\text{Minuten} = 11\text{Minuten} \Rightarrow W_A < W_B$ Sie empfehlen System A aufgrund der viel geringeren Wartezeit.

b) Fachbegriff: Monitoring ($\frac{1}{2}$P), spezielle Art der Implementierung: Software ($\frac{1}{2}$P) *1P*

c) Einsatz von Hardwaremonitoren ($\frac{1}{2}$P). Voraussetzung ist ein physischer Zugang zum System um diesen zu installieren ($\frac{1}{2}$P). *1P*

Leistungsbewertung der Gleitkommaarithmetik 7P

a) $i_{fpu} = \sum i_{typ} = (2.500 + 300 + 100 + 250) * 10^3 = 3.150.000$ *2P*
$cyc_{fpu} = (2.500 * 3 + 300 * 5 + 100 * 5 + 250 * 10) * 10^3 = 12.000.000$
$t_{fpu} = t_{cyc} * cyc_{fpu} \rightarrow 0,5ns * 12 * 10^6 = 6,0\,\text{ms}$ ($\frac{1}{2}$P)
$MIPS_{fpu} = \frac{i}{t*10^6} \rightarrow \frac{3.150.000}{6*10^{-3}\text{s}*10^6} = \frac{3.150}{6\text{s}} = 525\,\text{MIPS}$ ($\frac{1}{2}$P)
$i_{num} = i_{fpu} + (\frac{250.000}{5}) = 3.200.000$
$cyc_{num} = (2.500 * 3 + 1.750 + 100 * 5 + 250 * 10 + 250) * 10^3 = 12.500.000$
$t_{num} = t_{cyc} * cyc_{num} \rightarrow 0,5ns * 12,5 * 10^6 = 6,25\,\text{ms}$ ($\frac{1}{2}$P)
$MIPS_{num} = \frac{3.200.000}{6,25*10^{-3}\text{s}*10^6} = \frac{3.200}{6,25\text{s}} = 512\,\text{MIPS}$ ($\frac{1}{2}$P)

b) $Speedup_{num} = \frac{CPUtime_{fpu}}{CPUtime_{num}} = \frac{6\,\text{ms}}{6,25\,\text{ms}} < 1$ ($\frac{1}{2}$P) Nein, es lässt sich kein Speedup erzielen. Folglich schlagen Sie die bisherige Architektur ohne numerischen Beschleuniger vor. ($\frac{1}{2}$P) *1P*

c) Nimmt man die Gesamtlaufzeit als Bezugsgröße lässt sich die Leistungsfähigkeit unterschiedlicher Architekturen allgmein vergleichen, da man davon ausgeht, dass die Anzahl der Fließkommaoperationen nicht so stark von anderen Faktoren wie z.B. dem Compiler abhängt. Bezieht man sich auf die Ausführungszeit der Fließkommaoperationen, lassen sich unterschiedliche Fließkommaeinheiten hinsichtlich ihrer Effizienz beurteilen. Dies erlaubt keine Aussage über den restlichen Aufbau der Architektur und die Integration der Fließkommaeinheit. *1.5P*

d) Die reguläre FPU, da sie für die Berechnung nur $1,5 * 10^6$ Zyklen benötigt, was im Vergleich zum numerischen Beschleuniger ($1,75 * 10^6$ Zyklen) wenig ist. *0.5P*

e) Einzelne Befehle im Fließkommaformat sind unterschiedlich komplex und haben deshalb auch eine stark unterschiedliche Ausführungsdauer, so dass ein gemeinsam gebildeter Wert der Operationen (MFLOPS) nicht aussagekräftig ist. Ein einfacher add-Befehl hat den gleichen Einfluß auf die Metrik wie ein teurer diff-Befehl. *1P*

f) Speziell: $MFLOPS_{norm} = (2 * 0,6 + 4 * 0,4) * 300 = 840$ *1P*

Musterlösung 6: Prozessorarchitektur *9P*

Pipelining und Leistungsbewertung **4P**

a) Die Formel begründet sich durch die angenommene zeitliche Verarbeitungsdauer in einem Prozessor ohne Pipelining, es handelt sich lediglich um ein Modell. (0,5P) *1P*

Jeder Befehl müsste die gleichen Gatter und Flip-Flops innerhalb eines Taktes durchlaufen (mit Ausnahme der Pipeline-Register), weshalb die Taktperiode etwa k mal so lang sein müsste aufgrund der gleichen Verzögerung der einzelnen Elemente. (0,5P)

In der Realität kann es aber durchaus von Nutzen sein, eine längere Periode zu wählen und so die Anzahl an Pipelinestufen zu reduzieren, etwa um die Anzahl verworfener Befehle bei einem Pipeline-Flush zu verringern oder um Chipfläche/Transistoren einzusparen durch weniger Pipelineregister oder um den Energieverbrauch zu senken.

b) *2P*

$T_{\text{Takte}} = n + \text{Verzögerungen} + k - 1 = (1 + 0,04 * 2 + 0,04 * 3) * 100.000.000 + 7 - 1 = 1,2 * 100.000.000 + 6 = 120.000.006$ (0,5P)

$T_{\text{Sekunden}} = \frac{120.000.006}{2,4GHz} \approx \frac{12 * 10^7}{2,4 * 10^9} s = 5 * 10^{-2} s = 0,05s = 50ms$ (0,5P)

$S = \frac{T_{\text{seq}}}{T_{\text{pipe}}} = \frac{100.000.000 * 7}{120.000.006} \approx \frac{70}{12} = \frac{35}{6} \approx 5,8$ (0,5P)

Effizienz: $E = \frac{S(n)}{n} \approx \frac{\frac{70}{12}}{7} = \frac{5}{6} \approx 0,83$ (0,5P)

c) Das Problem liegt in Kontrollflussbefehlen, da diese nicht zweifach/doppelt ausgeführt werden können. (0,5P) Behebung ist dadurch möglich, dass die Ausführungseinheiten vervierfacht statt verdoppelt werden. (0,5P) *1P*

Sprungvorhersage **3P**

d) Der Korrelationsprädiktor ist in Abb. 1 dargestellt. *1,5P*

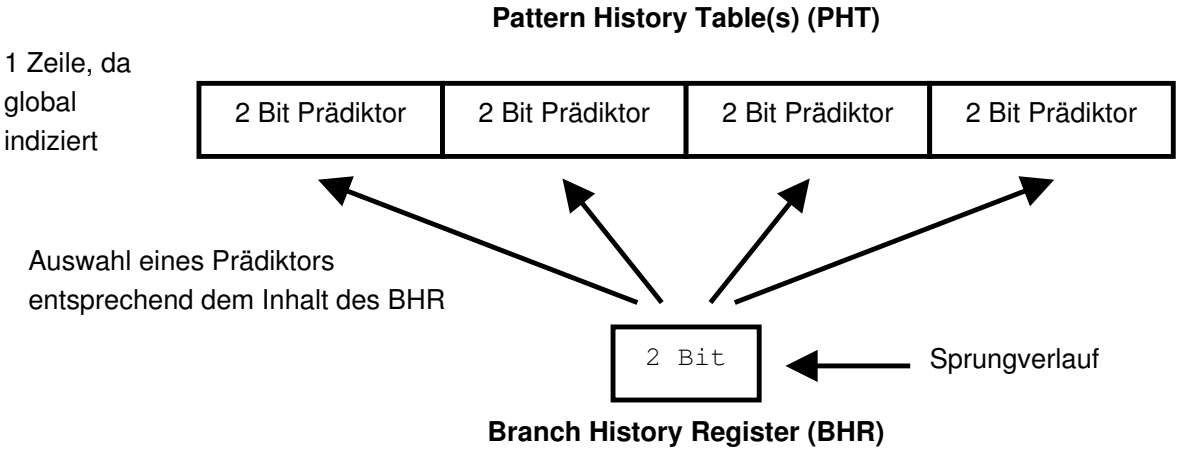


Abbildung 1: (2,2)-Korrelationsprädiktor mit global indizierter PHT

Je 0,5P für BHR, vier Prädiktoren, globale Indizierung mit einer Zeile.

e) Sprungverlaufstabelle: *1,5P*

Verlauf	Prädiktor	Vorhersage	Sprung	Aktualisierter Prädiktor
NT	(**T**,T)	**T**	NT	(NT, T)
NT	(**NT**,T)	**NT**	T	(T,T)
T	(T, **T**)	**T**	NT	(T,NT)

Superskalartechnik **2P**

f) Registerstatustabelle:

Register	1	2	3	4	5	6
Wert	–	–	(R3)	(R4)	(R5)	–
Gültig?	0	0	1	1	1	0
Reservation Station (RS#)	2	1	0	0	0	3

Reservation Stations:

RS#	Leer	In FU	Opcode	Ziel	Quelle1	Gültig1	RS1	Quelle2	Gültig2	RS2
1 Addierer	0	1	sub	2	(R4)	1	0	(R3)	1	0
2 Multiplizierer	0	1	mul	1	(R3)	1	0	(R5)	1	0
3 Dividierer	0	0	div	6		0	2	(R4)	1	0