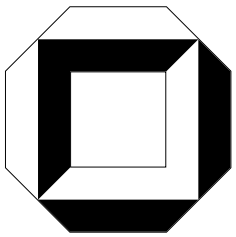


Universität Karlsruhe (TH)
Institut für Technische Informatik
Prof. Dr. Wolfgang Karl

Klausur Rechnerstrukturen
Wintersemester 2007/08
Musterlösung

Aushang der Ergebnisse: ab Zum VL-Beginn 2008

Musterlösung 1: Produktion und Betrieb

a) ($\frac{1}{2}$P) $dpw = \frac{\pi * (d_{wafer} * \frac{1}{2})^2}{a_{die}} - \frac{\pi * d_{wafer}}{\sqrt{2} * a_{die}} = A - B$ **1P**
($\frac{1}{2}$P) A: Gesamtfläche, B: Verschnitt

b) Der Anteil der Gesamtfläche wächst quadratisch ($\frac{1}{2}$P), der Verschnitt lediglich linear ($\frac{1}{2}$P). **1P**

c) $cost_{die} = \frac{cost_{wafer}}{dpw * yield_{die}} \leftrightarrow dpw = \frac{cost_{wafer}}{cost_{die} * yield_{die}}$ **1P**

d) $yield_{die} \sim f(a_{die}^4)$ **0,5P**

e) ($\frac{1}{2}$P) α ist die sogenannte Technologiekonstante. **1,5P**
($\frac{1}{2}$P) Sie ist ein Maß für den Fertigungsprozess (Schwierigkeit, Komplexität).
($\frac{1}{2}$P) Aufgrund des Formelwerks ergibt sich $\alpha > 0$.

(Hinweis: $\alpha = 0$ verbietet sich aufgrund des Bruchs, $\alpha < 0$ ist unlogisch, da so eine theoretische Ausbeute $> 100\%$ möglich wäre; konkret gilt $0 < \alpha \leq 1$ mit typischerweise $0,5 \leq \alpha \leq 0,9$.)

f) ($\frac{1}{2}$P) Ausfallwahrscheinlichkeit K: $1 - \Phi(K)$ **2P**

($\frac{1}{2}$P) Komplettausfall aller n Komponenten: $(1 - \Phi(K))^n$

($\frac{1}{2}$P) Damit Verfügbarkeit des Gesamtsystems: $1 - (1 - \Phi(K))^n)$

($\frac{1}{2}$P) Systemfunktion: $\bigwedge_{i=1}^{n} K_i$ bzw. $K_1 \wedge K_2 \wedge \ldots \wedge K_n$

g) ($\frac{1}{2}$P) Ein Totalausfall für das P_1-System ist gegeben für $(1 - \Phi(P_1))^2$. **2P**
(vgl. vorige Aufgabe)
($\frac{1}{2}$P) Somit muss für das P_2-System gelten: $P_2 > 1 - (1 - \Phi(P_1))^2)$.

Da $\Phi(P_2)$ für die gesamte Platte gilt ($\frac{1}{2}$P), ist die Spiegelung auf unterschiedliche Partitionen derselben Platte aus Fehlertoleranzsicht wirkungslos ($\frac{1}{2}$P).

h) ($\frac{1}{2}$P) Umtauschrecht$\sim$Initialphase (Inbetriebnahme) **1P**
($\frac{1}{2}$P) Garantie/Reparatur$\sim$Betriebsphase

Musterlösung 2: Hardwareentwurf *10P*

a) (1P) Verhalten: Was passiert? **3P**
(1P) Struktur: Wie passiert es?
(1P) Geometrie: Wo passiert es?

b) **1,5P**

```
entity xor is
  port(
    x, y: in  std_logic;
    z:    out std_logic
  );
end entity;
```

$\frac{1}{2}$P für Erfassung der Eingabesignale (Name, Richtung, Typ), $\frac{1}{2}$P für Erfassung des Ausgabesignals (Name, Richtung, Typ), $\frac{1}{2}$P für syntaktische Korrektheit (entity, port, Semikolon nach letztem Signal).

c) In der Architecture. **0,5P**

d) **1P**

```
architecture xor_arch of xor is
begin
  z<=a xor b;                              -- wer's gemerkt hat...

  z<=(a and not(b)) or (b and not(a));     -- wer nicht...

  z<='1' when a/=b else '0';               -- auch eine Möglichkeit

  z<='0' when a=b else '1';                -- so herum geht's auch

  z<='0' when a='0' and b='0' else
     '1' when a='0' and b='1' else
     '1' when a='1' and b='0' else
     '0' when a='0' and b='0';             -- für Vielschreiber
end architecture;
```

($\frac{1}{2}$P für korrekte Zuweisung, $\frac{1}{2}$P für korrekte Syntax.)

e) Es handelt sich um die Configuration ($\frac{1}{2}$P), welche zur Konfiguration (z.B. von Signalbreiten) bzw. der Festlegung verwendeter Architectures dient ($\frac{1}{2}$P). **1P**

f) In Alternative 1 wird für `flag` ein Speicherglied (Flipflop) synthetisiert ($\frac{1}{2}$P). Alternative 2 hingegen erzeugt ein rein kombinatorisches, nebenläufiges Signal ($\frac{1}{2}$P). **1P**

g) Das Vergleichsergebnis wird erst im Folgetakt sichtbar ($\frac{1}{2}$P), d.h. es wird entweder der Wert 1 (Aufwärtszähler) oder maximale Zählwert (Abwärtszähler) signalisiert ($\frac{1}{2}$P). **1P**

h) Das Signal `flag` verharrt auf dem Wert 0, da der Zähler selbst aufgrund der fehlenden Initialisierung im undefinierten Zustand verbleibt. **1P**

Musterlösung 3: Prozessorarchitektur **10P**

a) **1P**

- In superskalaren Prozessoren wird durch entsprechende Funktionseinheiten eine dynamische Parallelisierung erzielt. *0,5P*
- Bei VLIW- (und auch EPIC-) Prozessoren wird die Parallelisierung statisch durch den Compiler vorgenommen. *0,5P*

b) **1P**

- Familienbildung machbar, da von Parallelität auf Architekturebene entkoppelt *0,5P*
- Angabe der Einzelbefehlstypen in einem Wort *0,5P*

c) Echte Datenabhängigkeiten: **2P**

- R3: (1) → (2) *je 0,5P*
- R3: (1) → (4)
- R5: (3) → (4)

Datenkonflikte (Read-after-Write, RAW) treten zwischen der WB- und ID-Phase auf. *0,5P*

d) **1P**

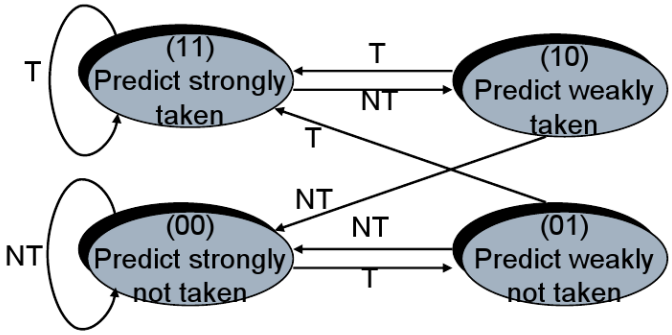
- Viele leichtgewichtige Pipelinestufen, dafür eine lange Pipeline (viele Pipelinestufen). *0,5P*
- Bei Steuerkonflikten (z.B. nach Sprüngen) werden viele Taktzyklen benötigt, um die Pipeline wieder zu füllen. *0,5P*

e) *je Fehler 0,5P Abzug* **4P**

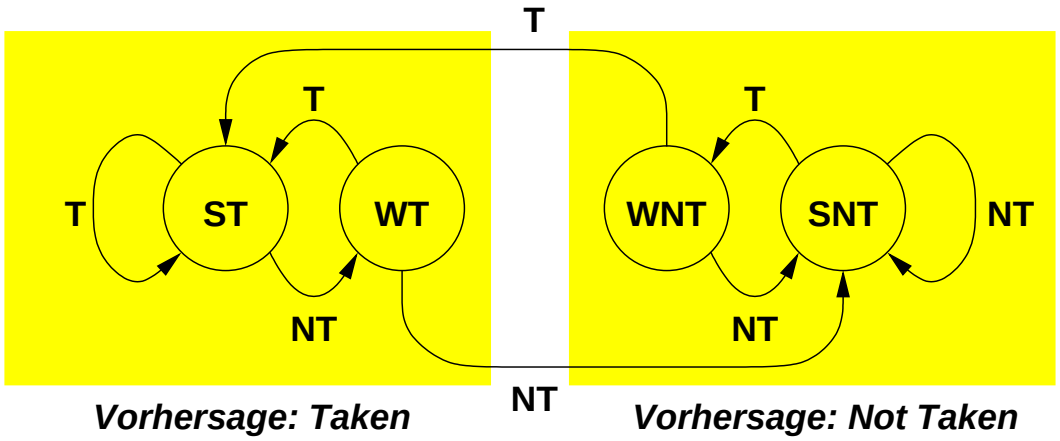
Ein-	Sprung 1				Sprung 2			
sprung	Prädiktor	Vorhersage	Sprung	P. neu	Prädiktor	Vorhersage	Sprung	P. neu
NT	(**NT**, NT)	NT	T	(T, NT)	(T, **NT**)	NT	T	(T, T)
–	(T, **NT**)	NT	NT	(T, NT)	(**T**, T)	T	NT	(NT, T)
–	(**T**, NT)	T	T	(T, NT)	(NT, **T**)	T	NT	(NT, NT)
–	(**T**, NT)	T	NT	(NT, NT)	(**NT**, NT)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

f) Hysteresezähler: **1P**

0,5P



oder



- Zustandsautomat mit 4 Zuständen: *0,5P*
  strongly taken (ST), weakly taken (WT), weakly not taken (WNT), strongly not taken (SNT)
- Zustandsübergänge:
  taken (T), not taken (NT)

Musterlösung 4: Parallelverarbeitung **10P**

a) Amdahls Gesetz: **1P**

$$T(n) = \underbrace{\frac{1}{n}}_{1} \underbrace{*T(1) * (1-a)}_{2} + \underbrace{T(1) * a}_{3}$$

- a: Anteil des Programmteils, der nur sequentiell ausgeführt werden kann *0,5P*
- Teil 1: Faktor, um den sich die Ausführungszeit des parallel ausführbaren Programmteils durch die Ausführung auf n Prozessoren verringert.
- Teil 2: Ausführungszeit des parallel ausführbaren Programmteils mit einem Prozessor.
- Teil 3: Ausführungszeit des nur sequentiell ausführbaren Programmteils. *0,5P*
- Teil 1 und 2 zusammengefaßt:
  Ausführungszeit des parallel ausführbaren Programmteils $1 - a$.

b) $T(n) = \lim_{n\to\infty} \left(\frac{1}{n} * T(1) * (1-a) + T(1) * a\right) = T(1) * a$ **1P**
$T(n) = T(1) * \frac{1}{32}$ *0,5P*
$S(n) = \frac{T(1)}{T(n)} = \frac{T(1)}{T(1)*\frac{1}{32}} = 32$ *0,5P*

c) $T(1) = 2048 \quad T(128) = 64$ **1P**
$S(n) = \frac{T(1)}{T(n)} \quad \rightarrow \quad S(128) = \frac{2048}{64} = 32$ *0,5P*
$E(n) = \frac{S(n)}{n} \quad \rightarrow \quad E(128) = \frac{S(128)}{128} = \frac{32}{128} = 0,25$ *0,5P*

d) Superlinearer Speedup: **1P**
$S > n, E > 1$ *0,5P*
Beispiele: *0,5P*

- paralleles Backtracking (depth-first search)
- Beim Programmlauf auf einem Rechner passen die Daten nicht in den Hauptspeicher des Rechners (häufiger Seitenwechsel), aber: bei Verteilung auf die Knoten des Multiprozessors können die parallelen Programme vollständig in den Cache- und Hauptspeichern der einzelnen Knoten ablaufen.

e) **1P**

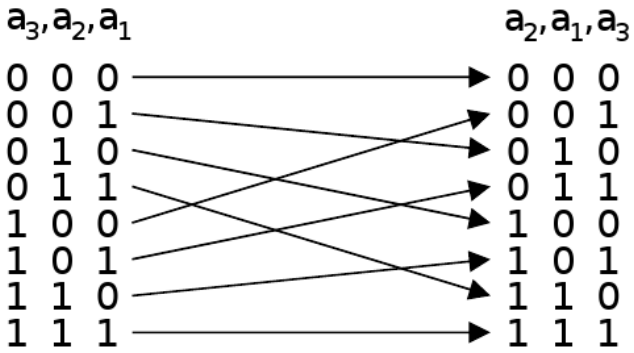
- Vektorrechner: Verarbeitung von großen Vektoren in einem Rechenwerk mit pipelineartig aufgebauten Funktionseinheiten unter Zuhilfenahme von Vektoroperationen. *0,5P*
- Nachrichtengekoppelte Parallelrechner: Verarbeitung von Daten aufgeteilt auf viele gleichartige Knoten, die verschiedne Befehlsfolgen auf verschiedenen Daten ausführen können. Nachrichtenaustausch über Verbindungsnetzwerk. *0,5P*

f) **1P**

- Vektorrechner: SIMD *0,5P*
  Nachrichtengekoppelte Parallelrechner: MIMD
- Weitere Klassen nach Flynn: SISD, MISD *0,5P*

g) Die Zweierschalter sind untereinander jeweils nach der Grundmuster der Mischpermutation verknüpft. **1P**

h) Mischpermutation: **1P**



i)
- Durchmesser: 6 *0,5P*
- Minimale Bisektionsbreite: 4 *0,5P*

j) **1P**

- Ausfallsicherheit: gibt keinen einzelnen Wurzelknoten, mehrere Wege möglich *0,5P*
  → hohe Ausfallsicherheit.
- Flaschenhals: keinen Wurzelknoten *0,5P*
  → kein direkter konstruktionsbedingter Flaschenhals, nur die Verbindungen zwischen den verschiedenen Ringen werden mehr belastet.

Musterlösung 5: Leistungsbewertung *10P*

Leistungsbewertung durch Befehlsmixe *Je Formel und Rechenweg je $\frac{1}{2}P$* *6P*

a) $\#Instruktionen\ i = \sum i_{typ} = (500 + 300 + 250 + 200) * 10^3 = 1.25 * 10^6$
$\#Zyklen\ c = \sum i_{typ} * c_{typ} = (500 * 1 + 300 * 5 + 200 * 10 + 250 * 4) * 10^3 = 5000000$
$Zykluszeit = t_{cyc} = \frac{t_{exec}}{c} = \frac{2.5*10^{-3}}{5*10^6} = \frac{2.5}{5*10^9} = 0.5ns$
$Taktfrequenz f = \frac{1}{t_{cyc}} = \frac{1}{0.5ns} = 2GHz$
$IPC = \frac{i}{t*f} = \frac{1.25*10^6)}{2.5*10^{-3}*2*10^9} = \frac{1{,}25*10^6}{5*10^6} = 0,25$
$MFLOPS = \frac{i_{Fließkomma}}{t_{Fließkomma}*10^6} = \frac{300*10^3}{5*300*10^3*0.5*10^{-9}*10^6} = \frac{10^3}{2.5} = 400MFLOPS$

Leistungsbewertung durch die SPEC Benchmark-Suite *4P*

b) $SPECratio = \frac{Referenzzeit_x}{Laufzeit_x}$ für einen beliebigen Benchmark x *$\frac{1}{2}P$*

c) Geometrisches Mittel über alle SPECratios des CINT2000 Benchmarks *$\frac{1}{2}P$*

d) SPECint2000 und SPECint_base2000 ($\frac{1}{2}$P) Die beiden Stufen unterscheiden sich in ihren Compileroptionen. In SPECint2000 sind auch aggressive und auf die Anwendung abgestimmte Optimierungen durch den Compiler erlaubt, bei SPECint2000 nur Standardoptimierungen. ($\frac{1}{2}$P) *1P*

e) (1P) $SPECrate_x = \frac{Sekunden\ pro\ Stunde}{Laufzeit_x\ von\ n_x\ Kopien\ auf\ Testsystem} * \frac{Referenzzeit_x}{längste\ Laufzeit}$ *2P*
Die Anzahl n_x ($\frac{1}{2}$P) der Kopien muss angegeben werden, da dieser Parameter frei gewählt werden kann. ($\frac{1}{2}$P)

Musterlösung 6: Speicherhierarchie *10P*

a) Wahrung der Konsistenz, d.h. die Daten, die im Hauptspeicher stehe, stimmen nicht immer mit den Daten in den einzelnen Caches überein. **1P**

b) Herstellung der Cache-Kohärenz, d.h. die Cache-Speicherverwaltung muß garantieren, daß nur auf die neusten Daten zugegriffen wird. **1P**

c) Es werden 2 weitere Zustandsbits pro Cachezeile ($\frac{1}{2}$P) und die Bus-Snooping-Logik mit entsprechenden Steuersignalen benötigt.($\frac{1}{2}$P) **1P**

d) MESI ist ein busbasiertes Protokoll; ein Bus ist in Systemen mit verteiltem gemeinsamen Speicher nicht vorhanden.($\frac{1}{2}$P) In solchen Systemen kommen verzeichnisbasierte Kohärenzprotokolle zum Einsatz. ($\frac{1}{2}$P) **1P**

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	-	-	-	-	-	-
1	rd 6	6/E					
2	rd 2			2/E			
1	rd 4		4/E				
3	rd 4		4/S			4/S	
2	rd 3				3/E		
3	wr 7						7/M
1	wr 4		4/M			4/I	
2	rd 7			7/E			7/S
3	wr 5					5/M	
1	rd 3	3/S			3/S		
3	wr 3	3/I			3/I		3/M
2	wr 7			7/M			7/M

f) Der neue Zustand heißt Owned. ($\frac{1}{2}$P) Durch den O-Zustand können Daten direkt von einem Cache zu anderen Caches transferiert werden, ein Zurückschreiben der Daten in den Hauptspeicher mit anschließendem Laden in die anderen Caches wird dadurch vermieden.(1P) Der Nachteil dieses 5. Zustands ist ein größerer Hardwareaufwand. Zur Speicherung von 5 Zuständen werden 3 Bits benötigt, bei 4 Zuständen genügen 2 Bits. ($\frac{1}{2}$P) **2P**