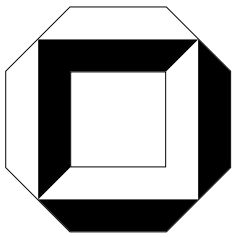

Universität Karlsruhe
Institut für Technische Informatik
Prof. Dr. Wolfgang Karl

Klausur Rechnerstrukturen
Wintersemester 2005/06
Musterlösung

Aushang der Ergebnisse: ab 8. Mai 2006

Musterlösung 1: Schaltungssynthese und Produktion *14P*

a) Die Schaltungsbeschreibung modelliert einen 4-Bit-Zähler, d.h. einen Zähler von 0-15 ($\frac{1}{2}$P). **3P**
`clk`: Löst beim Übergang von 0 nach 1 ($\frac{1}{2}$P) einen Zählschritt aus ($\frac{1}{2}$P)
`rst`: Löst bei 0-Pegel (=low-aktiv, $\frac{1}{2}$P) ein Rücksetzen des Zählers auf den Wert 15 aus ($\frac{1}{2}$P)
`count`: aktueller Zählerstand ($\frac{1}{2}$P)

b) Die Verwendung des Datentyps `std_logic` ist nicht zwingend: Es reicht die Verwendung des Datentyps `bit` aus ($\frac{1}{2}$P), da nur die logischen Zustände 0 und 1 benötigt werden ($\frac{1}{2}$P). **1P**

c) `count` hat die Signalrichtung *out* ($\frac{1}{2}$P), d.h. `count` kann nur geschrieben, aber nicht gelesen werden, wie es für die Addition notwendig ist ($\frac{1}{2}$P). Es ist zur Behebung ohne Änderung der Architecture die Signalrichtung in der Entity zu ändern ($\frac{1}{2}$P) und zwar auf *inout* oder *buffer* ($\frac{1}{2}$P). **2P**

d) A ist die theoretisch erzielbare maximale Anzahl von Dies ($\frac{1}{2}$P), B der durch die Einbeschreibung von Rechtecken entstehende Verschnitt ($\frac{1}{2}$P). **1P**

e) (1P) $dpw_{150} = \frac{\pi*(15cm*\frac{1}{2})^2}{2cm^2} - \frac{\pi*15cm}{\sqrt{2*2cm^2}}$ **2P**
$dpw_{150} = \pi * (\frac{15^2}{2^2*2} - \frac{15}{\sqrt{4}}) = \pi * (\frac{15^2}{4*2} - \frac{4*15}{4*2}) = \frac{225-60}{8}\pi = \frac{165}{8}\pi = 20\frac{5}{8}\pi$

(1P) $dpw_{250} = \frac{\pi*(25cm*\frac{1}{2})^2}{2cm^2} - \frac{\pi*25cm}{\sqrt{2*2cm^2}}$
$dpw_{250} = \pi * (\frac{25^2}{2^2*2} - \frac{25}{\sqrt{4}}) = \pi * (\frac{25^2}{4*2} - \frac{4*25}{4*2}) = \frac{625-100}{8}\pi = \frac{525}{8}\pi = 65\frac{5}{8}\pi$

f) (1P) $cost_{die} = \frac{cost_{wafer}}{dpw*yield_{die}}$ **3P**
(1P) $cost_{d150} = \frac{150}{25*\frac{1}{5}} = \frac{150}{5} = 30$
(1P) $cost_{d250} = \frac{250}{50*\frac{1}{5}} = \frac{250}{10} = 25$

g) Da die Herstellungskosten pro Die sinken und die Kosten für Test und Packaging pro IC gleichbleiben, muß sich somit die Gesamtausbeute verschlechtert haben. (1P) **2P**

Gemäß Aufgabenstellung und der Formel zur Berechnung der Kosten pro IC

$$cost_{IC} = \frac{cost_{die}+cost_{pkg}+cost_{test}}{yield_{final}}$$

folgend, können somit nur Beschädigung beim Packaging bzw. Testen des fertigen ICs Grund für das Absinken der Gesamtausbeute sein. (1P)

Musterlösung 2: Klassifikation und Leistungsanalyse von Rechensystemen *12P*

a) *(je $\frac{1}{2}$P)*: SISD (Single-Instruction, Single Data), SIMD (Single-Instruction, Multiple-Data), MIMD (Multiple-Instruction, Multiple-Data), MISD (Multiple-Instruction, Single-Data). **2P**

b) MIMD: ($\frac{1}{2}$P) unterschiedliche Prozessoren mit jeweils eigenständigen Programmen und Daten ($\frac{1}{2}$P) **1P**

c) (1P) $t_{CPU} = \frac{IC*CPI}{f_{CPU}} \rightarrow CPI = \frac{t_{CPU}*f_{CPU}}{IC}$ **3P**
(1P) $CPI_1 = \frac{2s*2*10^9*\frac{1}{s}}{\frac{3}{2}*10^9} = \frac{2*2*2}{3} = \frac{8}{3} = 2\frac{2}{3}$
(1P) $CPI_2 = \frac{2s*\frac{5}{2}*10^9*\frac{1}{s}}{5*10^9} = 1$

d) **3P**
(1P) $f_{CPU} = MIPS * 10^6 * CPI \rightarrow CPI = \frac{f_{CPU}}{10^6*MIPS}$
(1P) $CPI_1 = \frac{2*10^9}{10^6*\frac{3}{4}*10^3} = \frac{2*4}{3} = \frac{8}{3} = 2\frac{2}{3}$
(1P) $CPI_2 = \frac{\frac{5}{2}*10^9}{10^6*\frac{5}{2}*10^3} = 1$

e) Ein CPI-Wert von 1 erfordert eine ständig gefüllte und daher maximal ausgelastete Pipeline ($\frac{1}{2}$P). Dieser Zustand ist jedoch nur zwischen Einschwingphase und Abklingen der Pipeline erreicht (Anlaufen/Abklingen der Pipeline, $\frac{1}{2}$P). **1P**

f) Nicht erfaßt werden der Einfluß von Compiler ($\frac{1}{2}$P) und Betriebssystem ($\frac{1}{2}$P). Man bedient sich daher sogenannter Benchmarks ($\frac{1}{2}$P) wie beispielsweise die SPEC-Benchmark-Suite ($\frac{1}{2}$P). **2P**

Musterlösung 3: Rechnerarchitektur *10P*

a) (1P) while-Schleife: always not taken, weil Sprung nur bei Erfüllung der Abbruchbedingung genommen wird. **2P**
(1P) do/while-Schleife: always taken, weil Sprung nur bei Nichterfüllung der Abbruchbedingung nicht genommen wird.

b) Es ist der 2-Bit-Prädiktor zu wählen (1P). Im Gegensatz zum 1-Bit-Prädiktor tritt keine erneute Fehlvorhersage beim Wiedereintritt in die Schleife auf ($\frac{1}{2}$P), wartet jedoch mit einer vergleichbar geringen hardwaretechnischen Komplexität auf ($\frac{1}{2}$P). *(Der Korrelationsprädiktor erfordert zum einen erst eine "Einschwingphase", zum anderen erfordert er höheren technischen Aufwand.)* **2P**

c) Die Ursache liegt in den bedingten Sprüngen ($\frac{1}{2}$P), welche bei Fehlvorhersage zu einem Pipeline-Flush (mit Wiederanlauf) führen und somit eine bestmögliche Auslastung der Pipeline verhindern ($\frac{1}{2}$P). **3P**

Bei der bedingten Befehlsausführung ($\frac{1}{2}$P) wird Befehlen eine Bedingung zugeordnet ($\frac{1}{2}$P) und mit der Ausführung des jeweiligen Befehls ausgewertet ($\frac{1}{2}$P): Nur der Befehl liefert ein gültiges Ergebnis, dessen Bedigung erfüllt ist ($\frac{1}{2}$P).

d) Es stehen $2^{10} = 1024$ Sets zur Verfügung ($\frac{1}{2}$P). Jedes Set beinhaltet 2^m ($\frac{1}{2}$P) n-Bit-Prädiktoren ($\frac{1}{2}$P). **3P**

m beschreibt die Tiefe der Sprunghistorie ($\frac{1}{2}$P) und ist als m-Bit-Schieberegister ausgeführt ($\frac{1}{2}$P).

Für $m = 0$ verhält sich der (n,m)-Korrelationsprädiktor wie ein n-Bit-Prädiktor ($\frac{1}{2}$P).

Musterlösung 4: Parallelverarbeitung *12P*

a) VLIW/EPIC: statische Parallelisierung ($\frac{1}{2}$P) in Software ($\frac{1}{2}$P) durch den Compiler bzw. Programmierer ($\frac{1}{2}$P). **3P**

Superskalar: dynamische, d.h. zur Laufzeit stattfindende, Parallelisierung ($\frac{1}{2}$P) in Hardware ($\frac{1}{2}$P) durch Out-of-Order-Execution-Unit ($\frac{1}{2}$P).

b) Betroffen ist `U[i]` ($\frac{1}{2}$P), es handelt sich um eine echte Datenabhängigkeit ($\frac{1}{2}$P). **1P**

c) Die Abhängigkeit ist nicht schleifengetragen (loop-carried) ($\frac{1}{2}$P), daher kann die Schleife problemlos ausgerollt werden ($\frac{1}{2}$P). **1P**

d) ($\frac{1}{2}$P) X[2*i+4] $\rightarrow$ a=2, b=4; X[8*i+6] $\rightarrow$ c=8, d=6 **1P**
($\frac{1}{2}$P) GCD(a,c)=GCD(2,8)=2, (d-b)=6-4=2, 2 dividiert 2, Abhängigkeiten sind daher möglich.

e) Bei positivem Testausgang liefert der GCD-Test keine hinreichende Bedingung ($\frac{1}{2}$P). Beispielsweise werden keine Schleifengrenzen in den Test miteinbezogen ($\frac{1}{2}$P). **1P**

f) ($\frac{1}{2}$P) Komplexität $k = \frac{N*log_2 N}{2}$ **5P**
$N = 2^5 = 32 \rightarrow d = log_2 32 = 5$
($\frac{1}{2}$P) Diameter $d = log_2 N$
$N = 2^5 = 32 \rightarrow k = \frac{32*log_2 32}{2} = 16 * 5 = 80$

($\frac{1}{2}$P) A=2=(010), B=5=(101) $\rightarrow$ W=A xor B=111
($\frac{1}{2}$P) $W = 111$ bedeutet, es müssen alle 3 Dimensionen traversiert werden.

($\frac{1}{2}$P) Alternative 1: $(010) \rightarrow (000) \rightarrow (001) \rightarrow (101)$
($\frac{1}{2}$P) Alternative 2: $(010) \rightarrow (000) \rightarrow (100) \rightarrow (101)$
($\frac{1}{2}$P) Alternative 3: $(010) \rightarrow (011) \rightarrow (111) \rightarrow (101)$
($\frac{1}{2}$P) Alternative 4: $(010) \rightarrow (011) \rightarrow (001) \rightarrow (101)$
($\frac{1}{2}$P) Alternative 4: $(010) \rightarrow (110) \rightarrow (100) \rightarrow (101)$
($\frac{1}{2}$P) Alternative 4: $(010) \rightarrow (110) \rightarrow (111) \rightarrow (101)$

Musterlösung 5: Cache-Systeme *12P*

a) Bus Snooping ($\frac{1}{2}$P), setzt gemeinsamen Speicherbus voraus ($\frac{1}{2}$P) **1P**

b) **3P**

(b.1) In einer RISC-Pipeline sind Befehlshol- (IF) und (Daten-)Speicherzugriffsphase (MEM) typischerweise getrennt, d.h. können im Idealfall überlappend ausgeführt werden (1P). *2P*

Bei uniformem Befehls- und Datencache müßte der Cache-Speicher somit parallelen Zugriff durch beide Phasen unterstützen. Dies ist bei getrennten I- und D-Caches nicht notwendig (1P).

(b.2) Befehle und Daten verdrängen sich gegenseitig. *1P*

c) (1P) $\#m = (1-h) * z$ **4P**

($\frac{1}{2}$P) $\#m_I = (1-0,75) * 5000 = 0,25 * 5000 = 1250$
($\frac{1}{2}$P) $\#m_D = (1-0,8) * 3000 = 0,2 * 3000 = 600$

($\frac{1}{2}$P) $z_{L2} = \#m_I + \#m_D$
($\frac{1}{2}$P) $z_{L2} = 1250 + 600 = 1850$

($\frac{1}{2}$P) $h_{L1} = 1 - m_{L1} = 1 - \frac{z_{L2}}{z_I + z_D}$
($\frac{1}{2}$P) $h_{L1} = 1 - \frac{2000}{5000+3000} = 1 - \frac{2000}{8000} = 1 - 0.25 = 0.75$

d) *(je korrekte Zeile $\frac{1}{2}$P, je Fehler $\frac{1}{2}$P Abzug)* **4P**

		Prozessor/Cache 1		Prozessor/Cache 2	
Prozessor	Aktion	Line 1	Line 2	Line 1	Line 2
-	(init)	E/8	I/-	I/-	E/6
1	wr 2		M/2		
1	wr 8	M/8			
2	rd 10			E/10	
1	rd 6		S/6		S/6
2	wr 8	I/-		M/8	
2	rd 2				E/2
1	wr 2	M/2			I/-
2	rd 2	S/2			S/2