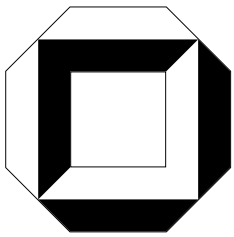

Universität Karlsruhe
Institut für Technische Informatik
Prof. Dr. Wolfgang Karl

Klausur Rechnerstrukturen
Sommersemester 2005
Musterlösung

Aushang der Ergebnisse: ab 5. September 2005

Musterlösung 1: Schaltungssynthese und Produktion *15P*

a) Die sogenannte Entity enthält Schnittstellendefinition des Moduls ($\frac{1}{2}$P), die Architecture die Funktionsbeschreibung ($\frac{1}{2}$P). *Hinweis: Die Configuration wird nicht zwingend benötigt sondern nur bei entsprechender Formulierung von Entity/Architecture.* *1P*

b) Von der Architecture ($\frac{1}{2}$P) können mehrere Versionen vorlegen für unterschiedliche Entwurfsphasen, z.B. abstrakte Beschreibung, RTL, Simulation ($\frac{1}{2}$P). *1P*

c) Ein Prozess benötigt eine sogenannte Sensitivity List ($\frac{1}{2}$P); eine Veränderung der in dieser Liste stehenden Signale löst die Abarbeitung der Prozessbeschreibung aus ($\frac{1}{2}$P). *1P*

d) VHDL kann sowohl für Simulation und Synthese verwendet werden ($\frac{1}{2}$P). Für eine synthetisierbare Beschreibung muss darauf geachtet werden, dass auch nur synthetisierbare Sprachkonstrukte verwendet werden ($\frac{1}{2}$P). *1P*

e) Die Formel hat die allgemeine Form A-B, daher $dpw = \frac{\pi*(d_{wafer}*\frac{1}{2})^2}{a_{die}} - \frac{\pi*d_{wafer}}{\sqrt{2*a_{die}}}$ *1P*

f) (1P) $dpw_{200} = \frac{\pi*(20cm*\frac{1}{2})^2}{2cm^2} - \frac{\pi*20cm}{\sqrt{2*2cm^2}}$ *2P*
$= \pi * (\frac{10^2}{2} - \frac{20}{\sqrt{4}}) = \pi * \frac{100-20}{2} = 40\pi$

(1P) $dpw_{300} = \frac{\pi*(30cm*\frac{1}{2})^2}{2cm^2} - \frac{\pi*30cm}{\sqrt{2*2cm^2}}$
$= \pi * (\frac{15^2}{2} - \frac{30}{\sqrt{4}}) = \pi * \frac{225-30}{2} = \frac{195}{2}\pi = 97,5\pi$

g) (1P) $yield_{die} = yield_{wafer} * (1 + \frac{defects/unit-area*a_{die}}{\alpha})^{-\alpha}$ *2P*
(1P) $yield_{die} = 0.75 * (1 + \frac{0.5*2}{1})^{-1} = \frac{3}{4} * \frac{1}{2} = \frac{3}{8}$

h) (1P) $cost_{die} = \frac{cost_{wafer}}{dpw*yield_{die}}$ *3P*
(1P) $cost_{d200} = \frac{200}{125*0.2} = \frac{200}{25} = 8$
(1P) $cost_{d300} = \frac{300}{250*0.2} = \frac{300}{50} = 6$

i) ($\frac{1}{2}$P) $cost_{ic} = \frac{cost_{die}+cost_{test}+cost_{pkg}}{yield_{final}}$ *3P*
(1P) $cost_{ic200} = \frac{8+1+0.75}{\frac{3}{4}} = \frac{9.75*4}{3} = \frac{39}{3} = 13$ (Euro)
(1P) $cost_{ic300} = \frac{6+1+0.75}{\frac{3}{4}} = \frac{7.75*4}{3} = \frac{31}{3} = 10\frac{1}{3}$ (Euro)
($\frac{1}{2}$P) Einsparung: $13 - 10\frac{1}{3} = 2\frac{2}{3}$ (Euro)

Musterlösung 2: Leistungsanalyse von Rechensystemen *11P*

a) Der CPI-Wert von Prozessor 1 ist <1, daher muss es sich um einen Prozessor mit parallelen Funktionseinheiten (superskalarer RISC, VLIW, EPIC) handeln. *1P*

b) Bei der Abarbeitung kommt es zu Pipeline-Konflikten ($\frac{1}{2}$P) bedingt durch Steuerflusskonflikte ($\frac{1}{2}$P), Datenflusskonflikte ($\frac{1}{2}$P) und Ressourcenkonflikte ($\frac{1}{2}$P). *2P*

c) (1P) Ausgangsformel: $CPU - Zeit = \frac{IC*CPI}{f}$ *3P*

(1P) Ansatz 1: $t_1 = \frac{1{,}5*10^6*\frac{3}{4}}{2{,}5*10^9}$, weitere Berechnung wie für t_1 untenstehend

(1P) Ansatz 2: $t_2 = \frac{1{,}5*10^6*\frac{3}{2}}{2{,}5*10^9}$, weitere Berechnung wie für t_2 untenstehend

Lösungsalternative (umständlich):
($\frac{1}{2}$P) Ausgangsformel 1: $CPI = \frac{Zyklen}{Befehle} \rightarrow Zyklen = CPI * Befehle$
($\frac{1}{2}$P) Ausgangsformel 2: $t_{run} = c * \frac{1}{f_{cpu}}$

Berechnung 1:

($\frac{1}{2}$P) $c_1 = \frac{3}{4} * \frac{3}{2} * 10^6 = \frac{9}{8} * 10^6 = 1125000$

($\frac{1}{2}$P) $t_1 = \frac{\frac{9}{8}*10^6}{\frac{5}{2}*10^9} = \frac{9}{20} * 10^{-3} = 450[\mu s]$

Berechnung 2:

($\frac{1}{2}$P) $c_2 = \frac{3}{2} * \frac{3}{2} * 10^6 = \frac{9}{4} * 10^6 = 2250000$

($\frac{1}{2}$P) $t_2 = \frac{\frac{9}{4}*10^6}{\frac{5}{2}*10^9} = \frac{9}{10} * 10^{-3} = 900[\mu s]$

d) ($\frac{1}{2}$P) Ausgangsformel: $f_{cpu} = MIPS * 10^6 * CPI \rightarrow MIPS = \frac{f_{cpu}}{CPI*10^6}$ *1,5P*

($\frac{1}{2}$P) $MIPS_1 = \frac{\frac{5}{2}*10^9}{\frac{3}{4}*10^6} = \frac{10}{3} * 10^3 = 3\frac{1}{3} * 10^3 \approx 3333,33 MIPS$

($\frac{1}{2}$P) $MIPS_2 = \frac{\frac{5}{2}*10^9}{\frac{3}{2}*10^6} = \frac{5}{3} * 10^3 = 1\frac{2}{3} * 10^3 \approx 1666,67 MIPS$

e) *entweder Alternative 1 (HW):* Pipeline-Stalling, d.h. das Anhalten der Pipeline, bis (zeitliche) Abhängigkeit aufgelöst ist. ($\frac{1}{2}$P) *1P*
oder Alternative 2 (SW): Einfügen von NOPs, so daß die (zeitliche) Abhängigkeit aufgelöst wird. ($\frac{1}{2}$P)

Beides vermindert den Durchsatz. ($\frac{1}{2}$P)

f) Es liegt ein Konflikt aufgrund einer echten Datenabhängigkeit vor ($\frac{1}{2}$P). Diese kann in Software – sofern möglich – durch Code-Umordnung bzw. Einfügen von NOPs *1,5P*

entschärft werden ($\frac{1}{2}$P); in Hardware kann sie durch eine Forwarding-Stufe (result forwarding) aufgelöst werden ($\frac{1}{2}$P).

g) Spekulation und Sprungvorhersage ($\frac{1}{2}$P) zur bestmöglichen Auslastung der Pipeline (Vermeidung von Wartezyklen bzw. Flushes) ($\frac{1}{2}$P). *1P*

Musterlösung 3: Rechnerarchitektur *14P*

a) Schleifenaus- und wiedereintritt lässt Vorhersage kippen ($\frac{1}{2}$P), daher zwei Fehlvorhersagen. Besser wäre hier tatsächlich die Verwendung einer "always taken"-Annahme. ($\frac{1}{2}$P) *1P*

b) Korrelationsprädiktoren ($\frac{1}{2}$P). Diese sind in der Lage, nicht nur die lokale Historie miteinzubeziehen sondern auch die globale ($\frac{1}{2}$P). *1P*

c) In der Schleife wird von 4 bis 0 gezählt ($\frac{1}{2}$P), wobei die ungeraden Zählwerte an eine Speicheradresse geschrieben werden ($\frac{1}{2}$P). *1P*

d) (je korrekte Zeile $\frac{1}{2}$P) *2P*

Durchlauf	R3	S1	R1	S2
1	0	T	3	T
2	1	NT	2	T
3	0	T	1	T
4	1	NT	0	NT

e) (je korrekte Zeile 1P) *4P*

	S1				S2			
Durchlauf	Präd.	Vhs.	Sprung	Präd. neu	Präd.	Vhs.	Sprung	Präd. neu
1	(NT, **T**)	T	T	(NT, T)	(T,**NT**)	NT	T	(T, **T**)
2	(NT, **T**)	T	NT	(NT,**NT**)	(**T**, T)	T	T	(T, T)
3	(NT, **NT**)	NT	T	(NT,**T**)	(T, **T**)	T	T	(T, T)
4	(NT, **T**)	T	NT	(NT,**NT**)	(**T**, T)	T	NT	(**NT**, T)

f) Da sich keine Korrelation zwischen den beiden Sprüngen ableiten lässt ($\frac{1}{2}$P), verhält er sich jeweils wie ein 1-Bit-Prädiktor, d.h. wechselt seinen Zustand mit jeder Fehlvorhersage ($\frac{1}{2}$P). *1P*

g) Verwendung eines 2-Bit-Prädiktors anstatt 1-Bit-Prädiktor ($\frac{1}{2}$P) verhindert jede zweite Fehlvorhersage ($\frac{1}{2}$P); Vergrößerung des BHR ($\frac{1}{2}$P) sorgt für bessere Korrelation ($\frac{1}{2}$P). *2P*

h) Beim 2-Bit-Prädiktor mit Sättigungszähler ($\frac{1}{2}$P) wird mit jeder Fehlvorhersage in den jeweiligen Nachbarzustand gewechselt ($\frac{1}{2}$P); beim 2-Bit-Prädiktor mit Hysteresezähler ($\frac{1}{2}$P) wird durch zwei Fehlvorhersagen direkt in den entsprechenden sicheren (*strongly*) Zustand gewechselt ($\frac{1}{2}$P). *2P*

Musterlösung 4: Parallelverarbeitung *10P*

a) MMX-Befehle fallen unter die SIMD-Kategorie ($\frac{1}{2}$P), da hier ein einzelner Befehl (single instruction) auf mehrere Datenbytes bzw. (Doppel)Wörter von Daten (multiple data) wirkt ($\frac{1}{2}$P). *1P*

b) ($\frac{1}{2}$P) X[i] $\rightarrow$ a=1, b=0; X[5*i+5] $\rightarrow$ c=5, d=5 *1P*
($\frac{1}{2}$P) GCD(a,b)=GCD(1,5)=1, (d-b)=5, 1 dividiert 5, Abhängigkeiten sind möglich.

c) GCD überprüft keine Schleifengrenzen ($\frac{1}{2}$P). Aufgrund Formulierung von Schleife und Zugriffsdistanz tritt hier keine loop-carried dependence auf ($\frac{1}{2}$P). *1P*

d) `k` bezeichnet die *dependence distance* ($\frac{1}{2}$P). Je größer `k` ist, umso mehr potentieller Parallelismus kann durch Loop-Unrolling ausgenutzt werden. ($\frac{1}{2}$P). *1P*

e) Durch die *dependence distance* von 7 existiert eine maximale Sequenz von 7 unabhängigen und potentiell parallel ausführbaren Instruktionen ($\frac{1}{2}$P). Im vorliegenden Beispiel ist die Parallelität aufgrund des Schleifenkonstrukts auf 5 begrenzt ($\frac{1}{2}$P). *1P*

f) Latenz setzt sich zusammen aus Kanalverzögerung (channel delay; Dauer der Belegung eines Kanals durch eine Nachricht), Schalt- oder Routingverzögerung (switching/routing delay; Zeit, einen Weg zwischen zwei Knoten aufzubauen) und Blockierungszeit (contention time; kollidierende Zugriffe auf die gemeinschaftliche Netzwerkressource). ($\frac{1}{2}$P je Begriff und Erklärung) *3P*

g) ($\frac{1}{2}$P) Komplexität=$\frac{N*log_2 N}{2}$ mit $N = 2^3 = 8 \rightarrow k = \frac{8*log_2 8}{2} = 4 * 3 = 12$ *2P*
($\frac{1}{2}$P) Diameter=$log_2 N$ mit $N = 2^3 = 8 \rightarrow d = log_2 8 = 3$
($\frac{1}{2}$P) A=5=(101), B=6=(110) $\rightarrow$ W=A xor B=011
($\frac{1}{2}$P) Alternative 1: $(101) \rightarrow (100) \rightarrow (110)$
Alternative 2: $(101) \rightarrow (111) \rightarrow (110)$

Musterlösung 5: Cache-Systeme *10P*

a) Weil hier der Datenaustausch über Knotengrenzen hinweg explizit über Kommunikation geschieht ($\frac{1}{2}$P) anstatt über gemeinsame Speicherbereiche ($\frac{1}{2}$P). *1P*

b) Erweiterung der Cache-Zeilen um 2 MESI-Zustandsbits ($\frac{1}{2}$P) erweitert und Unterstützung von (3) Kommunikationssignalen ($\frac{1}{2}$P). *1P*

c) *4P*

		Prozessor/Cache 1		Prozessor/Cache 2	
Prozessor	Aktion	Line 1	Line 2	Line 1	Line 2
-	(init)	E/8	I/-	I/-	E/6
2	wr 2			M/2	
1	wr 8	M/8			
1	rd 10		E/10		
2	rd 8	S/8			S/8
1	wr 8	M/8			I/-
2	wr 10		I/-		M/10
2	rd 18			E/18	
1	rd 18		S/18	S/18	

d) Ein Prozessor (P1) hält exklusive, modifizierte Kopie, auf die ein anderer Prozessor (P2) lesend zugreifen will ($\frac{1}{2}$P). *2P*

(d.1) Der Lesezugriff von P2 wird vom Cache-Controller von P1 erkannt und und unterbrochen. ($\frac{1}{2}$P)

(d.2) P2 schreibt modifiziertes Datum in Hauptspeicher zurück. ($\frac{1}{2}$P)

(d.3) P1 wird nach Rückschreiben per Retry aufgefordert, das Datum neu einzulesen. ($\frac{1}{2}$P)

e) *2P*

($\frac{1}{2}$P) $t_a = r_{H1} * t_{L1} + r_{M1} * (r_{H2} * t_{L2} + r_{M2} * t_{mem})$

(1P) $t_a = 0,75 * 2 + (1 - 0,75) * (0,8 * 5 + (1 - 0,8) * 200)$
$= \frac{3*2}{4} + \frac{1}{4} * (\frac{4*5}{5} + \frac{1*200}{5})$
$= \frac{3}{2} + \frac{1}{4} * (4 + 40) = 1,5 + 11 = 12,5(ns)$

($\frac{1}{2}$P) Beschleunigung=$\frac{200}{12,5} = \frac{400}{25} = 16$, d.h. der Speicherzugriff wird im Mittel um den Faktor 16 beschleunigt.