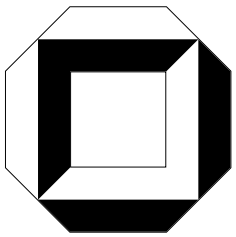


Universität Karlsruhe (TH)
Institut für Technische Informatik
Prof. Dr. Wolfgang Karl

Klausur Rechnerstrukturen
Sommersemester 2008
Musterlösung

Aushang der Ergebnisse: ab Ende September 2008

Musterlösung 1: Quantifizierung *10P*

a) ($\frac{1}{2}$P) $P_{total} = P_{switching} + P_{shortcircuit} + P_{static} + P_{leakage}$ **2,5P**
($\frac{1}{2}$P) $P_{switching}$: Laden/Schalten einer kapazitiven Last (Leitung)
($\frac{1}{2}$P) $P_{shortcircuit}$: Kurzzeitiger Kurzschluß bei Änderung des Ausgangspegels *(beide Transistoren leiten)*
($\frac{1}{2}$P) P_{static}: statischer Leistungsverbrauch
($\frac{1}{2}$P) $P_{leakage}$: Kriechströme

b) Prozessor 2 arbeitet in der gleichen Zeit doppelt so viele Befehle ab wie Prozessor 1, daher wird er doppelt so schnell getaktet sein ($\frac{1}{2}$P). Gemäß des Zusammenhangs $P \approx U^2 * f$ ($\frac{1}{2}$P) wird er somit auch (ungefähr) doppelt so viel Energie benötigen ($\frac{1}{2}$P). **1,5P**

c) **1,5P**

- $SPEC_{ratio} = \frac{Referenzzeit}{Laufzeit\ auf\ Testsystem}$ *0,5P*
- ($\frac{1}{2}$P) Aggressive und konservative Optimierung, *1P*
($\frac{1}{2}$P) Geschwindigkeit und Durchsatz

d) **1P**

- $MIPS = \frac{f}{CPI*10^6} \rightarrow CPI = \frac{f}{MIPS*10^6}$ *0,5P*
- Codegröße. *0,5P*
(MIPS/CPI erlauben nur die Beurteilung des Durchsatzes, nicht aber die Effizienz des Befehlssatzes.)

e) Die erzielbare Anzahl von Dies pro Wafern (dpw) wird überproportional ansteigen ($\frac{1}{2}$P). Dies liegt daran, dass der Beitrag der Wafer-Größe quadratisch in die dpw-Anzahl eingeht ($\frac{1}{2}$P), der Verschnitt jedoch nur linear ($\frac{1}{2}$P). **1,5P**

f) Die Die-Ausbeute steht über die Technologiekonstante α in direktem Zusammenhang mit der verwendeten Fertigungstechnologie ($\frac{1}{2}$P). Der so erzielte Wert wird nochmals durch die Wafer-Ausbeute skaliert ($\frac{1}{2}$P). **1P**

g) ($\frac{1}{2}$P) Optimierung von Test und Packaging gemäß **1P**
($\frac{1}{2}$P) $cost_{ic} = \frac{cost_{die}+cost_{test}+cost_{pkg}}{yield_{final}}$ bzw. $yield_{final} = \frac{cost_{die}+cost_{test}+cost_{pkg}}{cost_{ic}}$

Musterlösung 2: Hardwareentwurf *10P*

a) Funktional unabhängige Teilelemente sind unabhängig voneinander spezifiziert und realisiert ($\frac{1}{2}$P). Ein Befehlssatz ist orthogonal, wenn sich jeder ALU-Befehl mit jeder Adressierungsart kombinieren lässt ($\frac{1}{2}$P). **1P**

b) In einem symmetrischen System findet sich die mathematisch symmetrische Eigenschaften auch in einem entsprechend symmetrischen Entwurf wieder ($\frac{1}{2}$P), d.h. es ist zu erwarten, dass gleichartige Befehlstypen auch in gleichartiger Weise angewendet werden (z.B. SUB wie ADD) ($\frac{1}{2}$P). **1P**

c) Die Standardisierung folgt dem Grundsatz der Wiederverwendbarkeit ($\frac{1}{2}$P). Deren Zielsetzung ist außerdem eine Senkung der Kosten, also Sparsamkeit ($\frac{1}{2}$P). **1P**

d) ($\frac{1}{2}$P) Spezifikation: Entity (Schnitstellendeklaration) **1,5P**
($\frac{1}{2}$P) Realisierung: Architecture (Festlegung der Funktionalität)
($\frac{1}{2}$P) Test/Validierung: Testbench (Instantiierung des Modells mit Durchführung des Testszenarios)

e) ($\frac{1}{2}$P) Verhaltensspezifikation → High-level-Synthese → **2P**
($\frac{1}{2}$P) RT-Beschreibung → Logiksynthese →
($\frac{1}{2}$P) Gatterbeschreibung → Layout-Synthese (→ Geometriebeschreibung)
($\frac{1}{2}$P) Als Entwurfsmodell, wie am Entwurfsfluss ersichtlich, kommt das Top-Down-Modell zum Einsatz.

f) Die Zählerdekrementierung wird erst mit einer Verzögerung von einem Taktzyklus sichtbar ($\frac{1}{2}$P), darum wird das Signal `flag` tatsächlich den Zählerstand `0xfe` signalisieren ($\frac{1}{2}$P). **1P**

g) Variablen werden sofort aktualisiert ($\frac{1}{2}$P), d.h. es findet eine korrekte Signalisierung von `0xff` statt ($\frac{1}{2}$P). **1P**

h) `count` ist undefiniert ($\frac{1}{2}$P), der Wert `count-1` („undefiniert-1“) ist daher ebenfalls undefiniert ($\frac{1}{2}$P). Es fehlt eine Initialisierung (z.B. mittels Rücksetzsignal) ($\frac{1}{2}$P). **1,5P**

Musterlösung 3: Prozessorarchitektur *10P*

a) RISC: Befehle mit festem Aufbau von typ. 1 Zyklus Abarbeitungsdauer ($\frac{1}{2}$P), Steuerwerk typ. „hardwired“ / festverdrahtet implementiert ($\frac{1}{2}$P), Load/Store-Architektur ($\frac{1}{2}$P) **3P**

CISC: Befehlsaufbau und Ausführungszeit variabel ($\frac{1}{2}$P), Steuerwerk typ. mikroprogrammiert implementiert ($\frac{1}{2}$P), zahlreiche Adressierungsmodi (ideal: orthogonaler Aufbau) ($\frac{1}{2}$P)

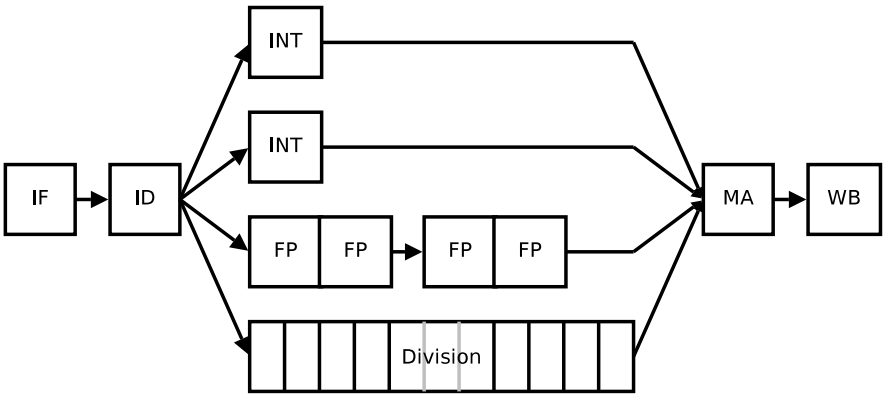
Hinweis: Die Ableitung „komplexer“ oder „einfacher“ Befehlssätze bzw. Architekturen aus der Namensgebung CISC/RISC ist irreführend. Es existierenden CISC-Architekturen, die ihrem Aufbau und Befehlssatz nach weit simpler sind als bestimmte RISC-Architekturen. Taktrate und Größte von Programmcode sind ebenfalls keine aus der CISC- bzw. RISC-Zuordnung ableitbaren Identifikatoren, sondern abhängig von der jeweiligen Implementierung.

b) **2P**

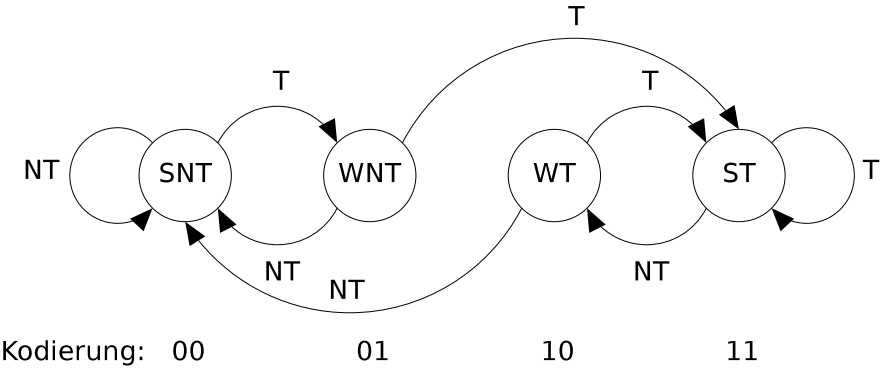
- Latenz: $k-1$=7, Durchsatz: 1 *0,5P*
- Die maximale Geschwindigkeit bestimmt sich aus der langsamsten Pipeline-Stufe. *0,5P*
- *($\frac{1}{2}$P für komplettes Formelwerk, $\frac{1}{2}$P für vollständige Berechnung)* *1P*
  $T = n + k - 1$
  $S = \frac{n*k}{T}$
  $S = \frac{993*8}{993+8-1} = \frac{8*1000-8*7}{1000} = \frac{7944}{1000} = 7,994$

c) *(mögliche Antworten, gewertet werden maximal 1,5 Punkte)* **1,5P**
Abhängigkeiten sind Eigenschaften des Programms ($\frac{1}{2}$P), Konflikte sind Resultate von Abhängigkeiten ($\frac{1}{2}$P), das Auftreten von Konflikten ist abhängig von der Pipeline-Organisation ($\frac{1}{2}$P). Es sind folgende Konfliktypen bekannt: Daten-, Steuer- und Ressourcen- bzw. Strukturkonflikt ($\frac{1}{2}$P)

d) *($\frac{1}{2}$P für grundsätzliche 5-stufige Pipeline, je $\frac{1}{2}$P für die einzelnen Einheiten)* **2P**



e) *(1P für korrekten Zustandsgraphen)* **1,5P**



Durch Verwendung der Hysterese fällt die Vorhersage bei zweimaliger Fehlvorhersage in den jeweils entgegengesetzten „starken“ (strongly) Zustand. ($\frac{1}{2}$P)

Musterlösung 4: Parallelverarbeitung *10P*

Leistungsfähigkeit von Multiprozessorsystemen: *4P*

a) ($\frac{1}{2}$P) Amdahls Gesetz: **1P**

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

($\frac{1}{2}$P) 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 Anteil des Programms, der nur sequentiell ausgeführt werden kann.*

b) **1P**
- ($\frac{1}{2}$P) Das Hinzufügen von weiteren Verarbeitungselementen führt zu einer kürzeren Gesamtausführungszeit, ohne dass das Programm geändert werden muss.
- ($\frac{1}{2}$P) Lineare Steigerung der Beschleunigung mit Anzahl der Knoten (bei einer Effizienz nahe 1).

c) *Das Problem selbst* muss skalieren (Problemgröße, Parallelisierbarkeit). **1P**

d) Das Verhalten heißt *superlineare Beschleunigung* bzw. *superlinearer Speedup* ($\frac{1}{2}$P) und widerspricht der Abschätzung $1 \leq S(n) \leq n$ ($\frac{1}{2}$P). **1P**

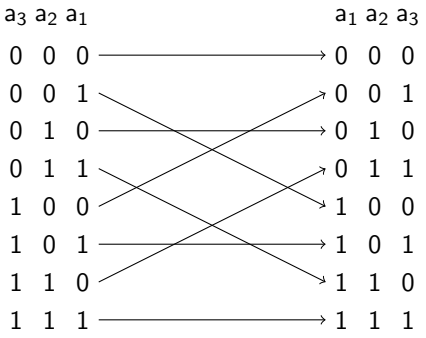
Parallelisierung und Parallelverarbeitung: *2P*

e) ($\frac{1}{2}$P) Verteilter Speicher: Nachrichtenorientiertes Programmiermodell **1P**
($\frac{1}{2}$P) Gemeinsamer Speicher: Shared-Memory-Programmiermodell

f) ($\frac{1}{2}$P) OpenMP: Parallelisierung erfolgt durch Compilerdirektiven **1P**
($\frac{1}{2}$P) MPI: Nutzung einer Bibliothek und Aufruf spezieller Funktionen (durch Programmierer)

Verbindungsnetze: *2P*

g) Kreuzpermutation: **1P**



h) Bei einem fehlertoleranten Netz muss *zwischen jedem Paar von Knoten* ($\frac{1}{2}$P) mindestens ein weiterer, redundanter Weg vorhanden sein ($\frac{1}{2}$P). **1P**

Vektorverarbeitung: *2P*

i) *(Jeweils $\frac{1}{2}P$ pro korrekter Antwort)* **2P**

- Vektor-Pipeline-Parallelität
- Mehrere Vektor-Pipelines in einer Vektoreinheit
- Vervielfachung der Pipelines
- Mehrere Vektoreinheiten

Musterlösung 5: Speicherhierarchie *10P*

a) $t_a = r_H * t_H + r_M * t_{Mem}$ **0,5P**

b) Alternative A: $t_a = 70\,\% * 10\,ns * 30\,\% * 100\,ns = 37\,ns$ ($\frac{1}{2}$P) **1,5P**
Alternative B: $t_a = 80\,\% * 15\,ns * 20\,\% * 100\,ns = 32\,ns$ ($\frac{1}{2}$P)

Entwurfsalternative B ist zu wählen, da hier das System eine bessere Leistung erzielt. ($\frac{1}{2}$P)

c) False-Sharing-Miss ($\frac{1}{2}$P) und True-Sharing-Miss ($\frac{1}{2}$P). **2P**

Bei einem True-Sharing-Miss wird der Miss durch einen Schreibzugriff auf ein Cache-Datum verursacht, welches von mehreren Prozessoren verwendet wird ($\frac{1}{2}$P).

Bei einem False-Sharing-Miss wird der Miss durch Zugriff auf ein Datum verursacht, das zwar nur von einem Prozessor verwendet wird, sich aber in einer Cache-Zeile befindet, in der von mehreren Prozessoren verwendete Daten liegen ($\frac{1}{2}$P).

d) Write-Back-Update- ($\frac{1}{2}$P) und Write-Back-Invalidation-Protokoll ($\frac{1}{2}$P). **2P**

Bei der Modifikation gemeinsamer Daten im Cache wird beim WB-Update-Protokoll das Datum in den anderen Cache(s) aktualisiert ($\frac{1}{2}$P) und beim WB-Invalidate-Protokoll invalidiert ($\frac{1}{2}$P).

e) *($\frac{1}{2}$P Abzug pro Fehler)* **3P**

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

f) Ja, die Verwendung des MOESI-Protokoll würde in diesem Fall zu einer Leistungssteigerung führen ($\frac{1}{2}$P): Durch die schnellen Cache/Cache-Transfers wird beim MOESI-Protokoll das ansonsten nötige Zurückschreiben von modifizierten Daten (Zeile 5 und 9) in den Hauptspeicher verhindert und spart somit 4 Hauptspeicherzugriffe ein ($\frac{1}{2}$P). **1P**

Musterlösung 6: Fehlertoleranz *10P*

a) *(Je vollständig korrekt zugeordnetem Fehlertyp $\frac{1}{2}P$)* **2P**

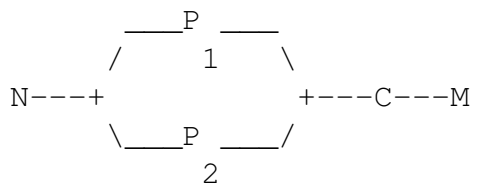
Entwurfsfehler	Betriebsfehler	Bedienungsfehler	Wartungsfehler	
☒	☐	☐	☐	Spezifikationsfehler
☒	☐	☐	☐	Implementierungsfehler
☒	☐	☐	☐	Dokumentationsfehler
☒	☐	☐	☐	Herstellungsfehler
☐	☒	☐	☐	Störungsbedingte Fehler
☐	☒	☐	☐	Verschleißfehler
☐	☒	☐	☐	Zufällige physikalische Fehler
☐	☐	☒	☐	Anschließen an unpassende Strom-/Spannungsversorgung
☐	☐	☒	☐	Fehlerhafter Gebrauch durch Anwender
☐	☐	☐	☒	Nichterkennen von eingetretenen Defekten aufgrund mangelhafter Inspektion
☐	☐	☐	☒	Austausch einer fehlerhaften Komponente durch ein inkompatibles Modell

b) Temporärer und permanenter Fehler. *(je $\frac{1}{2}P$)* **1P**

c) *(je 1P pro kompletter Antwort)* **3P**

- Fail-stop-System: Ausfälle sind nur Anhalteausfälle
- Fail-silent-System: Ausfälle sind nur Unterlassungsausfälle
- Fail-safe-System: Ausfälle sind nur unkritische Ausfälle

d) **1P**



e) $S = N \wedge (P_1 \vee P_2) \wedge C \wedge M$ **1P**

f) ($\frac{1}{2}$P) 2-Prozessorsystem nur bei Totalausfall funktionsuntüchtig. **1P**
Also gilt: $\Phi(2PS) = 1 - (1 - \Phi(P))^2$

($\frac{1}{2}$P) Somit ergibt sich für das Gesamtsystem: $\Phi(S) = \Phi(N) * \Phi(2PS) * \Phi(C) * \Phi(M)$

g) Gegenseitige Redundanz ($\frac{1}{2}$P) ermöglicht abgestuften Leistungsabfall bzw. graceful **1P**
degradation ($\frac{1}{2}$P).