Karlsruher Institut für Technologie

Institut für Technische Informatik

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

Prof. Dr. rer. nat. Wolfgang Karl

Klausur Rechnerstrukturen
Wintersemester 2014/15
Musterlösung

Voraussichtliche Veröffentlichung der vorläufigen Ergebnisse:
Ende März 2015

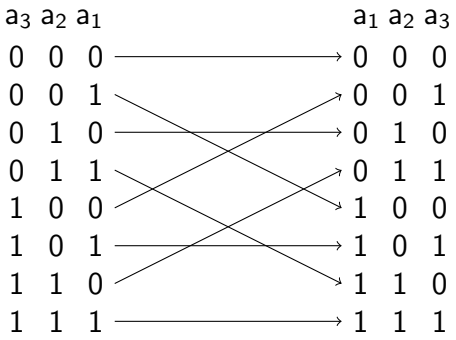
Musterlösung 1: Verbindungsstrukturen & Fehlertoleranz — 11 P

Verbindungsstrukturen — 3 P

a) Mögliche Antworten: *1 P*

- Fähigkeit, die wesentlichen Eigenschaften des Verbindungsnetzes auch bei beliebiger Erhöhung der Knotenzahl beizubehalten.
- Vergrößerung möglich ohne die wesentlichen Eigenschaften des Netzwerks zu verlieren.

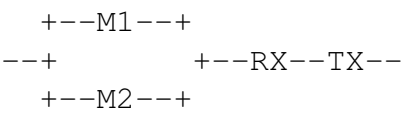
b) Kreuzpermutation: *1 P*



c) Bei einem fehlertoleranten Netz muss zwischen jedem Paar von Knoten (0,5P) mindestens ein weiterer, redundanter Weg vorhanden sein (0,5P) *1 P*

Fehlertoleranz — 8 P

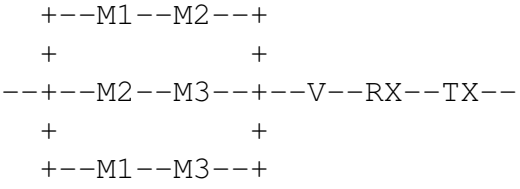
d) Zuverlässigkeitsblockdiagramm: *1 P*



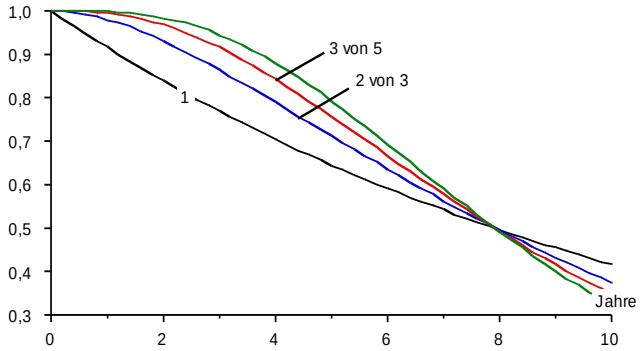
e) $S = (M1 \vee M2) \wedge RX \wedge TX$ *1 P*

f) SP parallel, d.h. $\varphi(M_p) = 1 - (1 - \varphi(M))^2$ *1 P*
Somit gilt für das Gesamtsystem: $\varphi(S) = \varphi(RX) * \varphi(TX) * (1 - (1 - \varphi(M))^2)$

g) Zuverlässigkeitsblockdiagramm: *2 P*



h) *2 P*



Begründung: Das 3-aus-5-System zeigt in der Anfangszeit eine höhere Funktionswahrscheinlichkeit als das 2-aus-3-System. Nach einiger Zeit (Schnittpunkt der beiden Graphen im Bild) kehrt sich dieser Effekt jedoch um. Anfangs sind die einzelnen Funktionswahrscheinlichkeiten noch hoch und bei einem 3-aus-5-System können bis zu 2 Komponenten ausfallen. Wenn die Funktionswahrscheinlichkeit einen bestimmten Punkt unterschreitet, ist ein 2-aus-3-System sicherer, da nur 2 Komponenten zum Betrieb notwendig sind.

i) Das 3-aus-5-System, da ein 2-aus-3-System erst nach ca. 8 Jahren sicherer ist. *1 P*

Musterlösung 2: Low-Power-Entwurf & Rechnerbewertung **10 P**

Low-Power-Entwurf **5 P**

a) Funktion f: *1 P*
$\mathbb{P}_{\neg A}(1) = \frac{2}{3}, \mathbb{P}_{\neg A \wedge B}(1) = \frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3}$
$\mathbb{P}_{(\neg A \wedge B) \vee C}(1) = 1 - \mathbb{P}_{(\neg A \wedge B) \vee C}(0) = 1 - (\frac{2}{3} \cdot \frac{3}{4}) = \frac{1}{2}$

b) Funktion f: *1 P*
$\mathbb{P}_{\neg A \wedge B} = 2 \cdot \frac{1}{3} \cdot \frac{2}{3} = \frac{4}{9}$
$\mathbb{P}_{(\neg A \wedge B) \vee C} = 2 \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}$

c) Formel: $P_{total} = P_{switching} + P_{shortcircuit} + P_{static} + P_{leakage}$ *1 P*

d) Formeln: *2 P*
$P_{switching} = C_{eff} \cdot V_{dd}^2 \cdot f$
$P_{shortcircuit} = I_{mean} \cdot V_{dd}$

Leistungsbewertung **5 P**

e) Antwort: *2 P*

- Warteschlangenmodelle
- Petrinetze
- Diagnosegraphen
- Netzwerkflussmodelle

f) Berechnung: *1 P*
$Spec_{ratio_A} = \frac{9\,s}{3\,s} = 3$
$Spec_{ratio_B} = \frac{4\,s}{6\,s} = \frac{2}{3}$

g) Berechnung: *2 P*
Anzahl der Instruktionen:
$IC = 1 \cdot 10^6 + 0,5 \cdot 10^6 + 0,3 \cdot 10^6 + 0,2 \cdot 10^6 = 2 \cdot 10^6$
Anzahl der Zyklen:
$C = 1 \cdot 10^6 \cdot 3 + 0,5 \cdot 10^6 \cdot 5 + 0,3 \cdot 10^6 \cdot 7 + 0,2 \cdot 10^6 \cdot 12 =$
$= 3 \cdot 10^6 + 2,5 \cdot 10^6 + 2,1 \cdot 10^6 + 2,4 \cdot 10^6 = 10 \cdot 10^6$
Ausführungszeit:
$T_{exe} = 10 \cdot 10^6 \cdot \frac{1}{2 \cdot 10^9}\, s = 5 \cdot 10^{-3}\, s = 5\, ms$
MIPS:
$MIPS = \frac{2 \cdot 10^6}{10^6 \cdot 5 \cdot 10^{-3}} = \frac{2000}{5} = 400$

Musterlösung 3: Parallelverarbeitung und Pipelining 10 P

Parallelverarbeitung 3 P

a) *3 P*

Ebene der Parallelität	Beispiel für eine Parallelarbeitstechnik
Programmebene	Grid-Computing, Cluster, Nachrichtenkopplung, Speicherkopplung (SMP/DSM),
Prozessebene	Grid-Computing, Cluster, Nachrichtenkopplung, Speicherkopplung (SMP/DSM), Grobkörniges Datenflußprinzip
Blockebene	Speicherkopplung (SMP/DSM), Grobkörniges Datenflußprinzip
Anweisungsebene	Befehlspipelining, Superskalar, VLIW, Überlappung von E/A- mit CPU-Operationen, Feinkörniges Datenflußprinzip
Suboperationsebene	Vektorrechnerprinzip, Feldrechnerprinzip, SIMD-Operationen,

Pipelining 7 P

b) **Laufzeit ohne skalarer Pipeline:** $T_{ohne} = n * k$ *1 P*
Laufzeit mit skalarer Pipeline: $T_{mit} = n + k - 1$

c) **Beschleunigung:** $S = \frac{n*k}{(k+n-1)}$ *1 P*
Begründung:
Diese Beschleunigung kann nur unter Annahme idealer Verhältnisse erzielt werden, d.h. in jedem Takt kann ein Befehl geholt bzw. beendet werden.

d) **Arten von Pipelinekonflikten:** *1,5 P*

- Strukturkonflikte
- Datenkonflikte
- Steuerkonflikte

e) **1. Strategie:** Anhalten der Pipeline (Pipeline stall) *1,5 P*
2. Strategie: Einfügen von Leerzyklen (No operations, NOPs), (Pipeline Bubble)
Resultat: Führt zu Leistungseinbußen

f) *2 P*

- **Einschränkung:** Obere Grenze des Durchsatzes, IPC <= 1 oder CPI >= 1
  Lösung: Nebenläufigkeit, parallele Pipelines

- **Einschränkung:** Ineffiziente Pipeline: Lange Latenzzeiten für eine Instruktion
  Lösung: diversifizierte, spezialisierte Pipelines
- **Einschränkung:** Pipeline Stall Strategie: Anhalten der Pipeline bewirkt, dass nachfolgende Befehle ebenfalls warten müssen
  Lösung:„Out-of-Order“ Strategie, verteilte Ausführungspipelines

Musterlösung 4: Fertigung und Hardware-Entwurf 10 P

Fertigungskosten 5 P

a) *3 P*

- Formel: $cost_{die} = \frac{cost_{wafer}}{dpw * yield_{die}}$
- Rechnung:
  $cost_{die_A} = cost_{die_B}$
  $\frac{cost_{wafer_A}}{dpw_A * yield_{die_A}} = \frac{cost_{wafer_B}}{dpw_B * yield_{die_B}}$
  $yield_{die_A} = \frac{cost_{wafer_B} * dpw_A * yield_{die_A}}{dpw_B * cost_{wafer_A}}$
  $yield_{die_A} = \frac{cost_{wafer_B} * yield_{die_A}}{cost_{wafer_A}}$
  $yield_{die_A} = \frac{400 * 0.75}{600} = \frac{2*3}{3*4} = \frac{6}{12} = 0.5$
- Antwort:
  Hersteller A muss mindestens eine Die-Yield von 0.5 garantieren, damit sich der Kauf seiner Wafer lohnt.

b) $yield_{die} = yield_{wafer} * (1 + \frac{dpua * A_{die}}{\alpha})^{-\alpha}$ *2 P*
$yield_{die}$: Ausbeute Die,
$yield_{wafer}$: Ausbeute Wafer,
$dpua$: Defekte pro Flächeneinheit,
α: Technologiekonstante (Maß für Komplexität bzw. Fertigungstechnologie)

Hardwareentwurf 5 P

c) Eine asynchrone Zuweisung wird unmittelbar an den Ausgang weitergeleitet, d.h. bei Änderung der Eingangssignale wird das Ausgangssignal nach einer gewissen Latenz geändert. Bei einer synchronen Zuweisung dagegen wird das Signal abhängig von einem Takt und den Eingangssignalen geändert. *1 P*

d) *2 P*

- Funktionalität: 4:1 Multiplexer
- Code:
  Möglichkeit 1:

```
if (input = "00") then
    output <= input0;
elsif (input = "01") then
    output <= input1;
elsif (input = "10") then
    output <= input2;
else
    output <= input3;
end if;
```

Möglichkeit 2:

```
case input is
    when "00" => output <= input0;
    when "01" => output <= input1;
    when "10" => output <= input2;
    when "11" => output <= input3;
    when others => output <= input0;
end case;
```

Hinweis: Es existieren noch weitere Lösungen

e) *2 P*

```
signal clk : std_logic;
signal reset : std_logic;
signal output : std_logic;

counter: process(clk, reset)
    variable counter_int : std_logic_vector(3 downto 0);
begin
    if (reset = '1') then
        output <= '0';
    elsif (rising_edge(clk)) then
        if (counter_int = "1010") then
            counter_int := "0000";
            output <= '1';
        else
            counter_int := counter_int + 1;
            output <= '0';
        end if;
    end if;
end process counter;
```

Hinweis: Es existieren noch weitere Lösungen

Musterlösung 5: Speicherhierarchie **10 P**

Cache-Kohärenzprotokoll MOESI **5 P**

a) *4 P*

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

b) Antwort: *1 P*

- Vorteil: Durch die Cache-Cache-Transfers werden 2 Hauptspeicherzugriffe gespart
- Nachteil:
  - Komplexere Logik notwendig
  - 3 Zustands-Bits nötig

Cache-Leistung **5 P**

c) Variante A: $0,75 \cdot 4\,ns + 0,25 \cdot (0,8 \cdot 20\,ns + 0,2 \cdot 100\,ns) =$ *2 P*
$= 3\,ns + 0,25 \cdot (16\,ns + 20\,ns) = 3\,ns + 0,25 \cdot 36\,ns = 3\,ns + 9\,ns = 12\,ns$

Variante B: $0,9 \cdot 8\,ns + 0,1 \cdot (0,75 \cdot 20\,ns + 0,25 \cdot 100\,ns) =$
$= 7,2\,ns + 0,1 \cdot (15\,ns + 25\,ns) = 7,2\,ns + 0,1 \cdot 40\,ns = 7,2\,ns + 4\,ns = 11,2\,ns$

Damit erreicht Variante B die kürzere Antwortzeit, weshalb ein Wechsel der Speicherhierarchie zu empfehlen ist.

d) Vorteil: Bei einem Miss muss nicht zuerst gewartet werden, bis Zugriff auf höhere Ebene durchgeführt wurde (geringere Zugriffszeit) *1 P*

Nachteil: Bei einem Hit in einem der Caches wurde zuvor trotzdem eine Anfrage an den Hauptspeicher gestartet, die dann abgebrochen wird. Der Bus wird also unnötigerweise blockiert.

e) Formel: $t_a = r_{H-L1} \cdot t_{H-L1} + (1 - r_{H-L1}) \cdot (r_{H-L2} \cdot (t_{H-L2} + t_{H-L1}) + (1 - r_{H-L2}) \cdot (t_{Mem} + t_{H-L1} + t_{H-L2}))$ *1 P*

f) Antwort: *1 P*

- Zeitliche Lokalität: Die zeitliche Lokalität besagt, dass ein in naher Vergangenheit verwendetes Datum in naher Zukunft erneut verwendet wird.
- Räumliche Lokalität: Die räumliche Lokalität besagt, dass auch auf die umgebende Nachbarschaft eines verwendeten Datums zugegriffen wird.

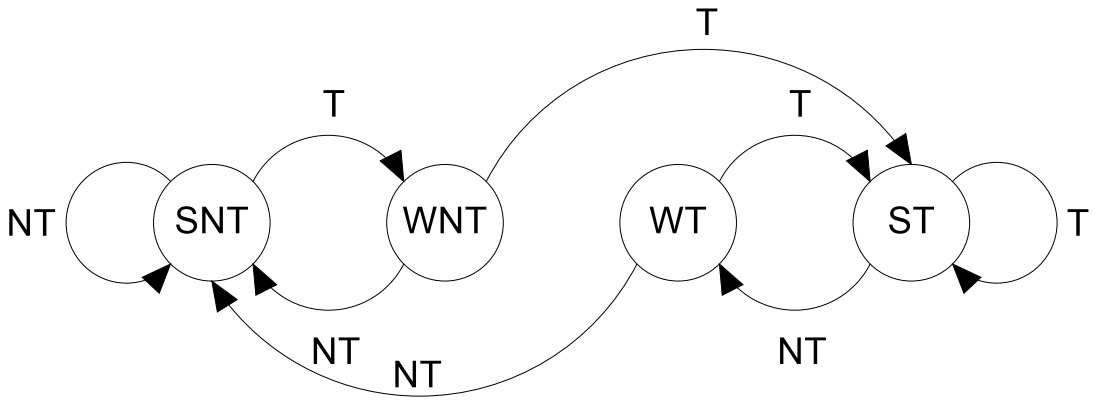
Musterlösung 6: Sprungvorhersage & VLIW **9 P**

Sprungvorhersage **5 P**

a) Tabelle: *4 P*

	Globaler	S1		Globaler	S2	
	Prädiktor	Vhs.	Sprung	Prädiktor	Vhs.	Sprung
1	(**T**, T)	T	NT	(**NT**, T)	NT	T
2	(T, **T**)	T	NT	(**T**, NT)	T	NT
3	(**NT**, NT)	NT	T	(T, **NT**)	NT	T
4	(T, **T**)	**T**	T	(T, **T**)	**T**	T

b) Zeichnung 2-Bit-Hysteresezähler *1 P*



VLIW **4 P**

c) VLIW-Prozessoren *4 P*

Slot 1	Slot 2
1) ld r3, [r1]	3) ld r4, [r2]
2) fpdiv f3, f1, f2	4) sub r7, r4, r5
5) ld r6, [r7]	6) fpadd f5, f3, f1
7) sub r8, r6, r7	8) st [r5], r7

System A

Integer	Gleitkomma	Load/Store
	2) fpdiv f3, f1, f2	1) ld r3, [r1]
	6) fpadd f5, f3, f1	3) ld r4, [r2]
4) sub r7, r4, r5		
		5) ld r6, [r7]
7) sub r8, r6, r7		8) st [r5], r7

System B

Entscheidung: Für System A, da das Programm trotz weniger Funktionseinheiten schneller abgearbeitet wird.