Karlsruher Institut für Technologie

Institut für Technische Informatik

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

Prof. Dr. rer. nat. Wolfgang Karl

Klausur Rechnerstrukturen Sommersemester 2016 Musterlösung

Voraussichtliche Bekanntgabe der vorläufigen Ergebnisse:
September 2016

Aufgabe 1: Sprungvorhersage und Fehlertoleranz 11 P

Sprungvorhersage 6 P

a) Die Sprungvorhersage wird üblicherweise in der Befehlsholphase (IF Phase) durchgeführt. *1 P*

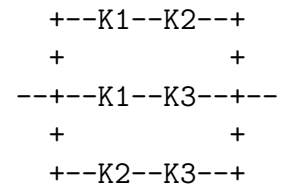
b) Bei der Prädiktion werden Sprungausgänge anhand des bisherigen Programmablaufs dynamisch vorhergesagt und Befehle entsprechend dieser Vorhersage spekulativ geladen und ausgeführt. Die Prädikation ist Compiler-gestützt, somit statisch, und vermeidet Sprungbefehle. Stattdessen sind Befehle mit einem Prädikat versehen und Ergebnisse werden nur gültig gemacht, wenn die Auswertung des Prädikatregisters Wahr ergibt. *2 P*

c) *3 P*

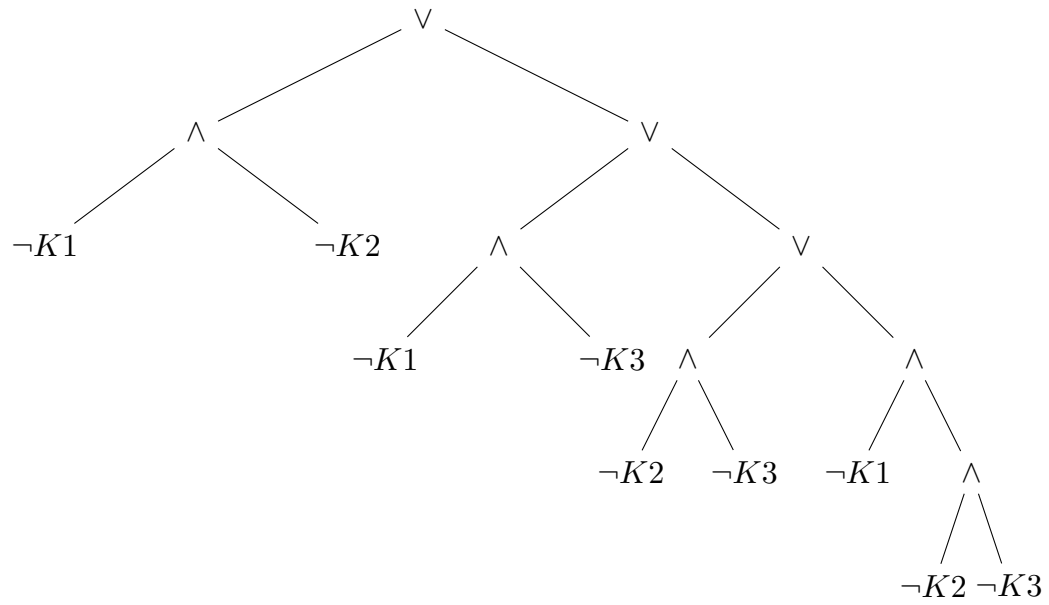
	Prädiktor 1	S1		Pärdiktor 2	S2	
		Vhs.	Sprung		Vhs.	Sprung
1	(**T**, T)	**T**	T	(**T**, T)	**T**	T
2	(T, **T**)	T	NT	(T, **T**)	**T**	T
3	(**T**, NT)	T	NT	(T, **T**)	T	NT
4	(**NT**, NT)	NT	T	(**T**, NT)	T	NT
5	(T, **NT**)	**NT**	NT	(**NT**, NT)	NT	T
6	(**T**, NT)	**T**	T	(T, **NT**)	**NT**	NT

Fehlertoleranz 5 P

d) Zuverlässigkeitsblockdiagramm: *2 P*



Fehlerbaum:



e) Es handelt sich um ein 2-aus-5 System. *0,5 P*

f) $V = \frac{MTBF}{MTBF+MTTR}$, wobei $MTBF$ die Mean Time Between Failures und $MTTR$ die Mean Time To Repair ist. *1 P*

g) Die Badewannenkurve beschreibt das Ausfallwahrscheinlichkeiten von Systemen über die Zeit. Die Kurve kann in die drei Phasen Frühphase, Betriebsphase und Spätphase aufgeteilt werden. *1,5 P*
Charakteristische Eigenschaften:

- Frühphase – Ausfallrate exponentiell abfallend (Initialausfälle)
- Betriebsphase – nahezu konstante Ausfallrate
- Spätphase – Ausfallrate exponentiell ansteigend (Alterungseffekte)

Aufgabe 2: VLIW und Superskalartechnik 9 P

VLIW 5 P

a) VLIW-Prozessoren *5 P*

Slot 1	Slot 2
2) ld r2, [r1]	3) ld r4, [r3]
1) fpmul f1, f3, f2	4) add r5, r2, r4
5) fpsub f4, f1, f5	6) ld r6, [r5]
7) sub r8, r6, r4	8) st [r1], r5
9) fpadd f6, f4, f1	10) st [r2], r8

System A

Integer	Gleitkomma	Load/Store
	1) fpmul f1, f3, f2	2) ld r2, [r1]
	5) fpsub f4, f1, f5	3) ld r4, [r3]
4) add r5, r2, r4	9) fpadd f6, f4, f1	
		6) ld r6, [r5]
7) sub r8, r6, r4		8) st [r1], r5
		10) st [r2], r8

System B

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

Superskalartechnik 4 P

b) Mit der Superskalatechnik wird die Einschränkung IPC (Instructions per Cycle) ≤ 1 aufgehoben. *1 P*

c) Der Rückordnungspuffer speichert die Befehle in Programmordnung und wird nach Befehlsausführung von der Rückordnungsstufe dazu verwendet, die Ergebnisse in Programmreihenfolge gültig zu machen. *1 P*

d) Mit der Registerumbenennung werden Ausgabe- und Gegenabhängigkeiten aufgelöst. *2 P*
Code-Beispiel:

```
add   r5, r2, r1
div   r5, r3, r4
```

Ausgabeabhängigkeit zwischen Befehl 1 und 2

Aufgabe 3: Caches **16 P**

a) *6 P*

Direct Mapped:

Durchlauf #	1						2						3						4					
Block Adresse	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
Cache-Zeile 0	0	0	0	0	4	4	0	0	0	0	4	4	0	0	0	0	4	4	0	0	0	0	4	4
Cache-Zeile 1	-	1	1	1	1	5	5	1	1	1	1	5	5	1	1	1	1	5	5	1	1	1	1	5
Cache-Zeile 2	-	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Cache-Zeile 3	-	-	-	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Fehlzugriffe	M	M	M	M	M	M	M	M			M	M	M	M			M	M	M	M			M	M

- Anzahl an Cache Fehlzugriffen:
  Im ersten Durchlauf ist der Cache leer und somit treten 6 Fehlzugriffe auf, in jedem weiteren Durchlauf treten jeweils nur Fehlzugriffe in Zeile 1 und 2 auf. Somit ist die Gesamtanzahl an Fehlzugriffen $= 6 + 4 * 9 = 42$ Fehlzugriffe.

Voll-assoziativ mit LRU:

Durchlauf #	1						2						3						4					
Block Adresse	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
Cache-Zeile 0	0	0	0	0	4	4	4	4	2	2	2	2	0	0	0	0	4	4	4	4	2	2	2	2
Cache-Zeile 1	-	1	1	1	1	5	5	5	5	3	3	3	3	1	1	1	1	5	5	5	5	3	3	3
Cache-Zeile 2	-	-	2	2	2	2	0	0	0	0	4	4	4	4	2	2	2	2	0	0	0	0	4	4
Cache-Zeile 3	-	-	-	3	3	3	3	1	1	1	1	5	5	5	5	3	3	3	3	1	1	1	1	5
Fehlzugriffe	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M

- Anzahl an Cache Fehlzugriffen:
  Im ersten Durchlauf entstehen aufgrund des leeren Caches 6 Fehlzugriffe. Durch die Verdrängungsstrategie können Blöcke nicht über mehrere Zyklen hinweg im Cache gehalten werden, so dass in jedem Durchlauf 6 Fehlzugriffe auftreten. Dadurch ist die Gesamtanzahl $= 6 * 10 = 60$ Fehlzugriffe.

Voll-Assoziativ mit LIFO:

Durchlauf #	1						2						3						4					
Block Adresse	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
Cache-Zeile 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cache-Zeile 1	-	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cache-Zeile 2	-	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Cache-Zeile 3	-	-	-	3	4	5	5	5	5	3	4	5	5	5	5	3	4	5	5	5	5	3	4	5
Fehlzugriffe	M	M	M	M	M	M				M	M	M				M	M	M				M	M	M

- Anzahl an Cache Fehlzugriffen:
  Im ersten Durchlauf entstehen aufgrund des leeren Caches 6 Fehlzugriffe. Ab da bleiben die ersten 3 Blöcke immer im Cache und Zeile 3 hält nacheinander 3, 4, und 5. Somit haben wir 3 Fehlzugriffe pro Durchlauf und die Gesamtanzahl ist $= 6 + 9 * 3 = 33$ Fehlzugriffe.

b) Die Anzahl an Cold Misses ist unabhängig von der Cache Organisation und Verdrängungsstrategie. Sie entstehen dadurch, dass der Cache initial leer ist und kann durch die Größe des Blockes unter Umständen minimiert werden. Ein Conflict Miss würde auftreten, wenn noch Platz im Cache vorhanden ist, aber aufgrund der Cache Organisation oder Verdrängungsstrategie trotzdem ein Block entfernt würde, auf den später zugegriffen wird. Dies geschieht im vorlegenden Fall nicht, daher sind alle nicht Cold Misses Capacity Misses. Capacity Misses entstehen dadurch, dass das Working Set größer ist als die Kapazität des Caches und dadurch zwangsläufig Daten/Instruktionen aus dem Cache verdrängt werden müssen. *6 P*

Hinweis: Aufgrund einer nicht klaren Unterscheidung in der Literatur, ob gleichzeitig ein Capacity Miss sowie Conflict Miss auftreten kann, sobald das Working Set größer als der Cache ist, werden für die Anzahl der Capacity Misses im Direct-Mapped Fall mehrere Lösungen akzeptiert.

Direct Mapped:

- Cold Misses: 6 Fehlzugriffe
- Capacity Misses: 36/(0) Fehlzugriffe
- Conflict Misses: 36 Fehlzugriffe

Voll-assoziativ mit LRU:

- Cold Misses: 6 Fehlzugriffe
- Capacity Misses: 54 Fehlzugriffe
- Conflict Misses: 0 Fehlzugriffe

Voll-assoziativ mit LIFO:

- Cold Misses: 6 Fehlzugriffe
- Capacity Misses:27 Fehlzugriffe
- Conflict Misses: 0 Fehlzugriffe

c) *4 P*

Proz.	Aktion	Proz. 1		Proz. 2		Proz. 3	
		Zeile 1	Zeile 2	Zeile 1	Zeile 2	Zeile 1	Zeile 2
-	init	-	-	-	-	-	-
3	wr 2					2/M	
1	rd 2	2/S				2/S	
2	rd 3			3/E			
3	rd 4						4/E
2	rd 2				2/S		
3	wr 2	2/I			2/I	2/M	
3	rd 3			3/S		3/S	
1	rd 1	1/E					
1	wr 3		3/M	3/I		3/I	
3	wr 6					6/M	

Aufgabe 4: Hardware-Entwurf, Fertigungskosten und VHDL — 14 P

Hardware-Entwurf — **6 P**

a) **Technische Herausforderungen:** *2 P*

- **Leistungs -und Energieaufnahme**
  Die Leistungsaufnahme in modernen Mikroprozessoren besteht aus 2 Anteilen: einem Statischen und einem Dynamischen. Insbesondere der statische Anteil ist rasant gestiegen durch sogenannte Leckströme. Die statische Leistungsaufnahme ist insbesondere bei Caches hoch. Der dynamische Anteil wird vor allem in den Prozessorkernen verbraucht.
- **Verlässlichkeit**
  In einem Prozessor können auch aufgrund der Reduzierung der Spannung verschieden Arten von *Faults* auftreten. Diese müssen beim Entwurf der Hardware berücksichtigt werden.
- **Signalverzögerung/Gatterlaufzeit**
  Die Signallaufzeit auf Verbindungen (Wires) skaliert nicht im gleichen Maße wie die Schaltgeschwindigkeit von Transistoren bei kleineren Strukturbreiten. Insbesondere bei Cache-Strukturen führt es dazu, dass die Zugriffszeit bei größeren Caches zunimmt.
- **Entwurfskomplexität**
  Aufgrund der hohen Komplexität des Entwurfs bekommt die Verifikation von Hardware einen immer höheren Stellenwert.
- **Limit der Miniaturisierung**
  CMOS Strukturen lassen sich nicht mehr ohne Weiteres in kleineren Strukturen herstellen.

b) **Befehlssatzarchitekturen:** *3 P*

- **Reduced Instruction Set Computer (RISC)**
  Die Idee hinter RISC ist eine Befehlsatzarchitektur, die aus primitiven Instruktionen besteht und weitere nur hinzugefügt werden, wenn dies aus Leistungsaspekten sinnvoll ist. Für eine einfache Dekodierung und Pipelining sowie kurzen Zykluszeiten haben die Instruktionen eine feste Länge und das Format besitzt eine sehr reguläre Struktur.
- **Complex Instruction Set Computer (CISC)**
  Ein Hauptziel von CISC war, dass der Instruktionsfluss so gut wie möglich kodiert wird. Somit entstand eine variable Instruktionslänge sowie ein variables Format. Des Weiteren sollte sowohl der Speicher als auch der Cache besser ausgenutzt werden. Zudem sollte die Komplexität der Programmierung reduziert werden, da zu dieser Zeit der Einsatz von Assemblersprachen

Standard war.

c) **Vorgehensweisen Hardware-Entwurf** *1 P*

- Bottom-Up Entwurf
- Top-Down Entwurf

Fertigungskosten **2 P**

d) Kosten des Dies, Kosten für das Testen des Dies, Kosten für das Packaging und den abschließenden Test. *1 P*
Formel: $cost_{IC} = \frac{cost_{die} + cost_{ie-test} + cost_{packaging}}{Y_{final}}$

e) Die erzielbare Anzahl von Dies pro Wafer (dpw) wird überproportional ansteigen. Dies liegt daran, dass der Beitrag der Wafer-Größe quadratisch in die dpw-Berechnung eingeht, der Verschnitt jedoch nur linear. *1 P*

VHDL **6 P**

e) Entity-Beschreibung: 0,5 P Benennung, 0,5 P Syntax *6 P*
Architecture-Beschreibung: 1 P Benennung, 1 P Syntax/Semantik

```
ENTITY lauflicht is
     PORT(
         clk : in std_logic;
         led : out std_logic_vector(7 downto 0)
     );
END lauflicht;

ARCHITECTURE behavioral of lauflicht is
    COMPONENT clockdivider
    PORT(
        clkin_in : IN std_logic;
        clkdv_out : OUT std_logic;
    );
    END COMPONENT;

    signal clk_int : std_logic;
    signal sr : std_logic_vector (7 downto 0) := "10000000";

BEGIN
    Inst_cd : clockdivider
    PORT MAP(
        clkin_in => clk,
        clkdv_out => clk_int,
    );

    shift: PROCESS(clk_int)
    BEGINN
        IF (rising_edge(clk_int)) THEN
            sr <= sr(0) & sr(7 downto 1);
        END IF;
    END PROCESS
    led <= sr;
END ARCHITECTURE;
```

Aufgabe 5: Low-Power-Entwurf und Leistungsbewertung — 10 P

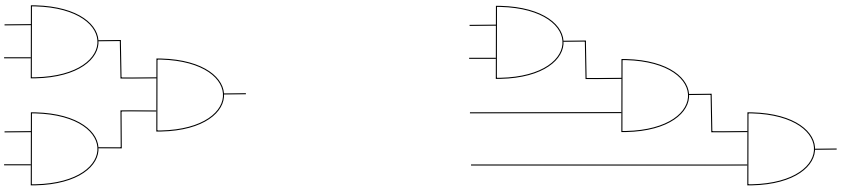
Low-Power-Entwurf — 7 P

a) *2 P*

- Schaltwahrscheinlichkeit: $\mathbb{P}_{schalt} = 2 * \mathbb{P}(1) * (1 - \mathbb{P}(1))$ (0,5 Punkte)
- Leistungsaufnahme: $P_{total} = P_{switching} + P_{shortcircuit} + P_{static} + P_{leakage}$ (0,5 Punkte)
- Bestandteil: $P_{leakage}$ (0,5 Punkte)
- Begründung: Zunehmende Integrationsdichte (0,5 Punkte)

b) *5 P*

Verschaltung 1 Verschaltung 2



- Durchlaufzeit: Verschaltung 1 wird eine geringere Durchlaufzeit zeigen als Verschaltung 2. Begründung: Durch das parallele Durchschalten von Gattern entfallen schaltbedingte Wartezeiten. (0,5 Punkte)
- Leistungsaufnahme: Verschaltung 1 wird eine höhere Leistungsaufnahme zeigen als Verschaltung 2. Begründung: Bedingt durch die höhere Schaltwahrscheinlichkeit wird diese Verschaltung häufiger schalten. (0,5 Punkte)
- Schaltwahrscheinlichkeit (mit Berechnung):

Verschaltung 1 (2 Punkte):

$$\mathbb{P}_{links}(1) = \left(\frac{1}{2}\right)^2 = \frac{1}{4}$$
$$\mathbb{P}_{linksSchalt} = 2 * \frac{1}{4} * \frac{3}{4} = \frac{3}{8}$$
$$\mathbb{P}_{rechts}(1) = \left(\frac{1}{4}\right)^2 = \frac{1}{16}$$
$$\mathbb{P}_{rechtsSchalt} = 2 * \frac{1}{16} * \frac{15}{16} = \frac{15}{128}$$
$$\mathbb{P}_{schalt} = \frac{3}{8} + \frac{3}{8} + \frac{15}{128} = \frac{111}{128}$$

Verschaltung 2 (2 Punkte):

$$\mathbb{P}_{links}(1) = \left(\frac{1}{2}\right)^2 = \frac{1}{4}$$
$$\mathbb{P}_{linksSchalt} = 2 * \frac{1}{4} * \frac{3}{4} = \frac{3}{8}$$
$$\mathbb{P}_{mitte}(1) = \frac{1}{4} * \frac{1}{2} = \frac{1}{8}$$
$$\mathbb{P}_{mitteSchalt} = 2 * \frac{1}{8} * \frac{7}{8} = \frac{7}{32}$$
$$\mathbb{P}_{rechts}(1) = \frac{1}{8} * \frac{1}{2} = \frac{1}{16}$$
$$\mathbb{P}_{rechtsSchalt} = 2 * \frac{1}{16} * \frac{15}{16} = \frac{15}{128}$$
$$\mathbb{P}_{schalt} = \frac{3}{8} + \frac{7}{32} + \frac{15}{128} = \frac{91}{128}$$

Verschaltung 1 zeigt damit eine höhere Schaltwahrscheinlichkeit als Verschaltung 2.

Leistungsbewertung **3 P**

c) Gesetz von Little: $k = \lambda t$, alternativ: $Q = \lambda\omega$ (1 Punkt) *2 P*
Erklärung: k = Mittlere Auftragszahl, λ = Durchsatz, t = Antwortzeit, Q = Mittlere Warteschlangenlänge, ω = Wartezeit (1 Punkt)

d) Voraussetzung: Statistisches Gleichgewicht(Rate der eingehenden Aufträge = Rate der abgehenden Aufträge) (0,5 Punkte) *1 P*
Begründung: Die Warteschlange läuft sonst über oder leer. (0,5 Punkte)