Musterlösungen zur
Nachklausur
Kurs 01709 „Technische Informatik III“

vom 18.09.2004

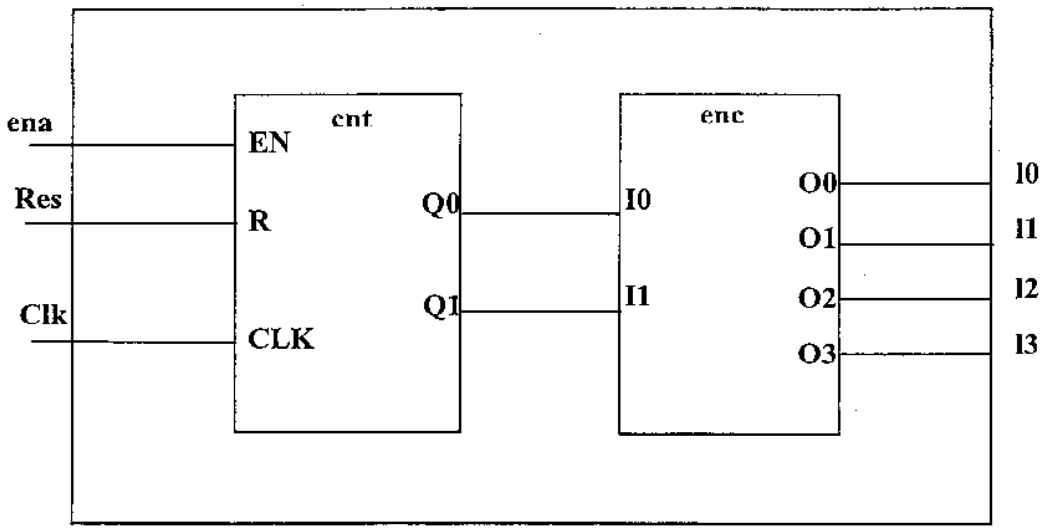
©2004 FernUniversität in Hagen

Aufgabe 1 (Grundlagen) 30 *Punkte*

Es soll ein VHDL-Design entwickelt werden, welches ein Eingangssignal zählt und mit Hilfe von 4 LEDs, LED1 bis LED4 den aktuellen Zählerstand anzeigt:

- Der Eingangstakt **clk** soll von 0 bis 3 gezählt werden (steigende Taktflanken).
- Bei Erreichen des Zählerstands 3 wird bei der nächsten Taktflanke automatisch auf 0 zurückgesetzt.
- Das Signal **res** dient zum asynchronen Zurücksetzen des Zählers (bei R='1').
- Das Signal **ena** dient zum Ein- (bei EN='1') und Ausschalten des Zählers.
- Die Ausgabesignale **I0-I3** dienen zum Ansteuern der LEDs.

Es wurde folgendes Schaltungsdesign entworfen:



4P

a) Geben Sie die Entities für den Zähler **cnt** und für den Encoder **enc** an!

```
ENTITY cnt IS
      PORT (EN, R, CLK          : IN bit;
            Q0, Q1              : OUT bit);
END cnt;

ENTITY enc IS
      PORT (I0, I1              : IN bit;
            O0, O1, O2, O3      : OUT bit);
END enc;
```

16P

b) Entwerfen Sie eine Verhaltensbeschreibung für den Zähler **cnt**.

Hinweise:

- steigende Taktflanken lassen sich mit folgender Bedingung abfragen: *if (CLK'event and CLK = '1') then...*
- Der Zähler lässt sich am besten mit Hilfe eines Zustandsautomaten beschreiben.
  Variable: state = 0, state = 1...
- Die Ausgabe (Q0, Q1) ist binär codiert.

```
ARCHITECTURE behaviour OF cnt IS
SIGNAL state : integer RANGE 0 TO 4 := 0; --zaehler(zu)stand
BEGIN
   PROCESS (CLK, R)
   BEGIN
      IF (EN = '1') THEN
         IF (R = '1') THEN
               state <= 0;
               ELSIF (CLK'event AND CLK = '1') THEN
                     state <= state+1;
               END IF;
               IF (state = 4) THEN
               state <= 0;
               END IF;
         END IF;
   END PROCESS;

   --nebenläufige Zuweisungen
  Q0 <=         '1' WHEN state = 1 ELSE
                '1' WHEN state = 3 ELSE
                '0';
  Q1 <=         '1' WHEN state = 2 ELSE
                '1' WHEN state = 3 ELSE
                '0';
END behaviour;
```

10P

c) Die 4 LEDs sollen unter folgenden Bedingungen angeschaltet werden:

LED1 (l0) = An, wenn Zählerstand = 0;
LED2 (l1) = An, wenn Zählerstand = 1;
LED3 (l2) = An, wenn Zählerstand = 2;
LED4 (l3) = An, wenn Zählerstand = 3;

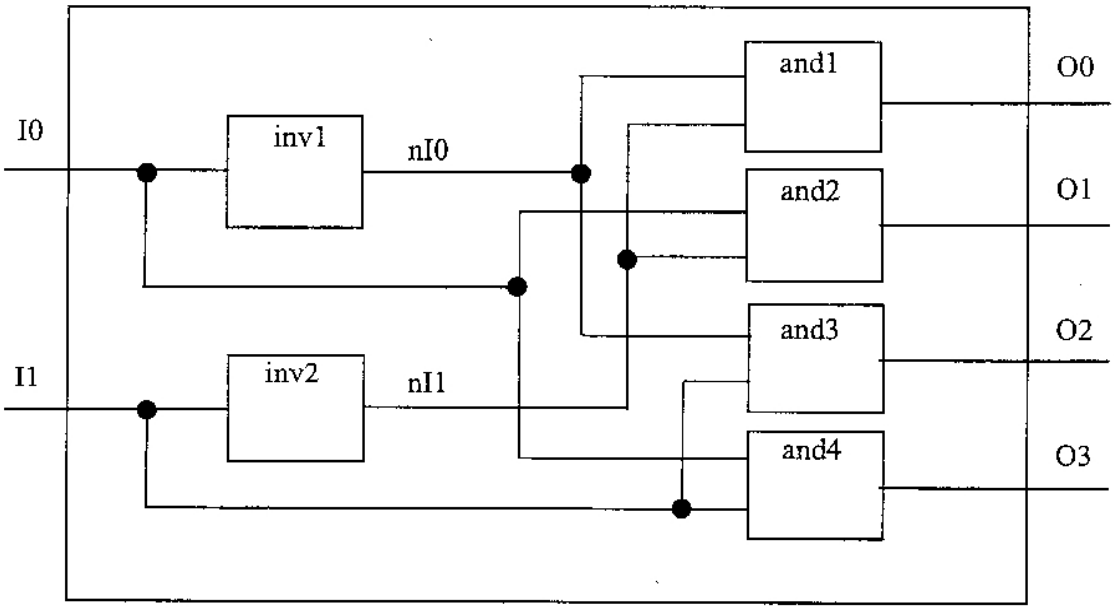
Für die Entwicklung von stehen zwei Typen von Bausteinen (Inverter und logisches AND) zur Verfügung:

```
ENTITY and IS
     PORT (a, b : IN  bit;
           y    : OUT bit);
END and;

ENTITY inv IS
     PORT (a    : IN  bit;
           y    : OUT bit);
END inv;
```

Geben Sie eine Strukturbeschreibung für den Encoder **enc** an!

Folgendes Schaltbild realisiert einen korrekten Encoder:



```
ARCHITECTURE structural OF enc IS
    COMPONENT inv                          -- Komponentendeklaration
        PORT (a     : IN bit;
              y     : OUT bit);
    END COMPONENT;
    COMPONENT and
        PORT (a, b  : IN bit;
              y     : OUT bit);
    END COMPONENT;
    SIGNAL nI0, nI1 : bit;

BEGIN
    inv1 : inv                             -- Komponenteninstanziierung
        PORT MAP (I0, nI0);
    inv2 : inv
        PORT MAP (I1, nI1);
    and1 : and
        PORT MAP (nI0, nI1, O0);
    and2 : and
        PORT MAP (I0, nI1, O1);
    and3 : and
        PORT MAP (nI0, I1, O2);
    and4 : and
        PORT MAP (I0, I1, O3);
END structural;
```

Aufgabe 2 (Grundlagen) 3 *Punkte*

1P

a) Wofür steht die Abkürzung EPIC?

Explicitly Parallel Instruction Computing

2P

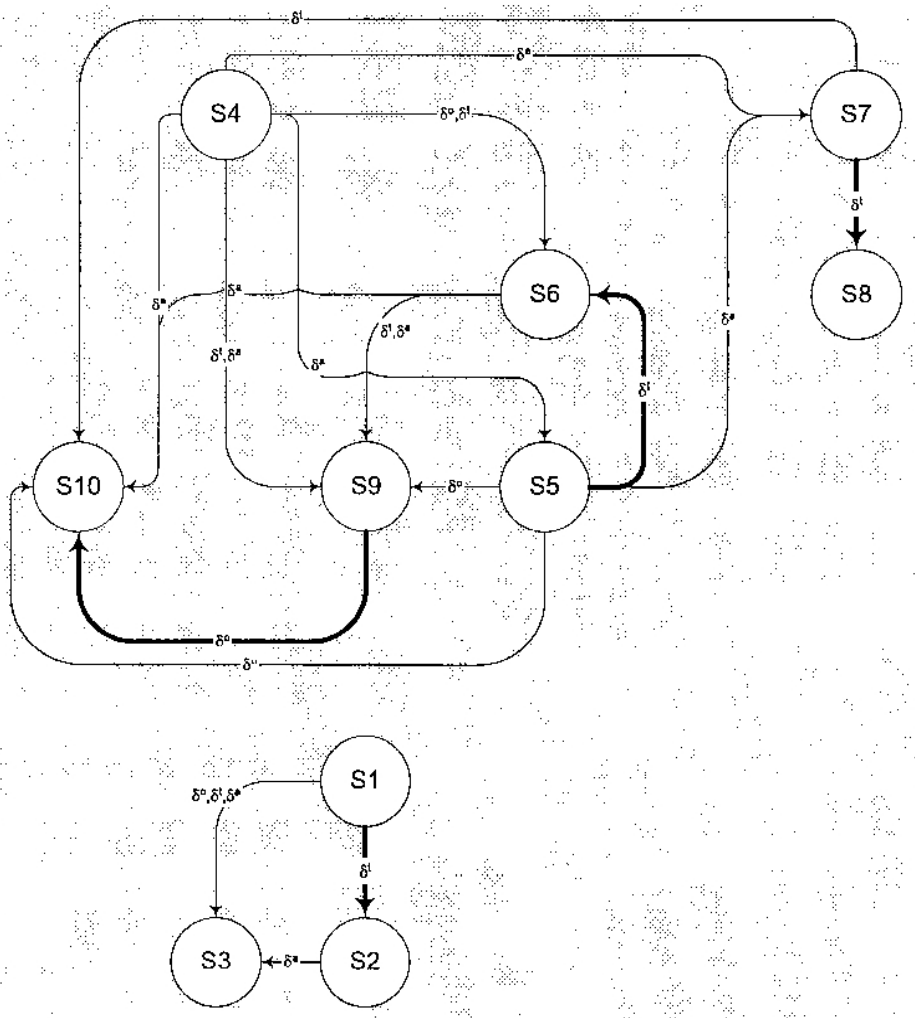
b) Was unternehmen der Programmierer, bzw. der Compiler, um parallele Ausführung bei EPIC zu erreichen?
Compiler und/oder Programmierer teilen der Mikroarchitektur explizit mit, welche Befehle in einem oder mehreren Befehlsworten parallel ausgeführt werden können.

Aufgabe 3 (Pipelining) *23 Punkte*

Gegeben sei folgendes DLX-Programm:

```
; Pipeline-Konflikt Testprogramm
main:
S1:         add         r1,r1,8
S2:         beqz        r1,Finish
S3:         add         r1,r1,3
S4:         multf       f2,f1,f4
S5:         multf       f1,f8,f4
S6:         multf       f2,f1,f2
S7:         multf       f4,f11,f4
S8:         multf       f5,f12,f4
S9:         addf        F1,F2,F3
S10:        movf        F1,F4
Finish:
S11:        trap        0
```

Geben Sie den Datenabhängigkeitsgraphen für o.a. Programm an. Kennzeichnen Sie im Abhängigkeitsgraphen die unterschiedlichen Arten dabei durch die Bezeichnungen δ^t, δ^a, δ^o wie im Kurstext. S11, bzw. Trap #0 soll dabei nicht in die Untersuchung mit eingehen.



Die dicken Linien kennzeichnen Pipeline-Konflikte die Stalls in der DLX-Pipeline verursachen.

Aufgabe 4 (Cache) 12 *Punkte*

5P

a) Erläutern Sie die Begriffe:
   - n-Wege-assoziative Organisation,
   - voll-assoziative Organisation und
   - direkt abbildende Organisation

n-Wege assoziative Caches stellen eine Mischform aus vollassoziativen und direkt abbildenden Caches dar. Ein bestimmter Hauptspeicherblock kann bei dieser Organisation in eine Gruppe von n verschiedenen Cache-Einträgen übertragen werden, womit ein Mittelweg zwischen einfacher Implementierung und akzeptabler Flexibilität beschritten wird. Pro Hauptspeicher-Eintrag gibt es jeweils n Cache-Einträge, in denen der Eintrag abgelegt werden kann.

Voll-assoziative Caches arbeiten nach dem Prinzip des Assoziationsspeichers. Jeder Hauptspeichereintrag der Größe m Byte, der an einer Adresse beginnt, die ohne Rest durch die Eintragsgröße teilbar ist, kann bei voll-assoziativer Organisation in jedem Cache-Eintrag abgelegt werden.

Direkt abbildende Caches bilden das genaue Gegenteil zu vollassoziativen Caches und sind nach folgendem Grundprinzip organisiert:

Jeder Hauptspeicher-Eintrag wird auf genau einen Cache-Eintrag abgebildet.

1P

b) **Was versteht man unter „LRU"?**

Die Least-Recently-Used-Strategie. Eine Cache-Ersetzungsstrategie. Derjenige der n Einträge, auf den am längsten nicht mehr zugegriffen wurde, wird ersetzt.

2P

c) **Wie kann man sie implementieren?**

Je mehr Einträge zur Verfügung stehen, desto aufwendiger wird es, diese Strategie zu implementieren. Man kann beispielsweise pro Eintrag e_k einen Alterungszähler verwenden, der bei jedem Zugriff auf einen der anderen n-1 Einträge hochgezählt wird, und auf 0 gesetzt wird, wenn direkt auf e_k zugegriffen wird. Der Eintrag mit dem höchsten Zählerstand ist der Ersetzungskandidat. Denkbar sind auch entscheidungsbaumbasierte Pseudo-LRU-Verfahren, die ohne Zähler auskommen, aber in manchen Fällen nicht den LRU-Eintrag finden.

2P

d) Diskutieren Sie (wenn Vorhanden) die Vor- und Nachteile großer Cache-Line-Größen.

Vorteil: Bei einem Cache-Hit liegt der nächste Befehl/ bzw. das nächste Datum meist schon vor.

Nachteil: Bei einem Cache-Miss muss trotzdem die komplette Line übertragen werden

→ Zeitverlust.

2P

f) Wann bezeichnet man eine Cache-Speicherverwaltung als **cache-kohärent?**

Eine Cache-Speicherverwaltung heißt **cache-kohärent,** falls ein Lesezugriff immer den Wert des zeitlich letzten Schreibzugriffs auf das entsprechende Speicherwort liefert.

Aufgabe 5 (MESI-Protokoll) 9 Punkte

Gegeben sei ein speichergekoppeltes Multiprozessorsystem mit zwei Prozessoren, die über einen Bus mit einem globalen Speicher verbunden sind. Zur Wahrung der Cache-Kohärenz wird das MESI Protokoll verwendet. Die Caches der beiden Prozessoren haben je eine Größe von drei Cache-Lines, die genau ein Speicherwort aufnehmen können. Die Caches werden von der niedrigsten Cache-Line aufwärts gefüllt, falls noch freie Lines zur Verfügung stehen. Im Falle eines voll besetzten Caches werden sie nach der Strategie LRU (least recently used) überschrieben. Ergänzen Sie die folgende Tabelle, wobei die Buchstaben die Zustände M = exclusive modified, E = exclusive unmodified, S = shared unmodified, I = invalid repräsentieren. Die Zahlen hinter den Aktionen und Zuständen bezeichnen Speicheradressen.

Prozessor	Aktion	Cache 1			Cache 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
-	-	E/8	E/12	I/-	E/6	I/-	I/-
1	read 18						
2	write 18						
1	write 8						
2	read 10						
1	read 10						
2	read 8						
1	write 8						
2	write 10						

Prozessor	Aktion	Cache 1			Cache 2		
		Line 1	Line 2	Line 3	Line 1	Line 2	Line 3
-	-	E/8	E/12	I/-	E/6	I/-	I/-
1	read 18			E/18			
2	write 18			I/-		M/18	
1	write 8	M/8					
2	read 10						E/10
1	read 10			S/10			S/10
2	read 8	S/8			S/8		
1	write 8	M/8			I/-		
2	write 10			I/-			M/10

Aufgabe 6 (Multiprozessorsysteme) 23 Punkte

3P

a) Bei der Nachrichtenübertragung zwischen nicht benachbarten Sender- und Empfängerknoten müssen in den Zwischenknoten Puffer eingerichtet sein, um Nachrichten, Pakete oder zumindest einzelne Übertragungseinheiten von Paketen zwischenzuspeichern. Bei der Zwischenspeicherung muss der Knoten dafür sorgen, dass sein Puffer nicht überläuft. Dies geschieht durch eine zusätzliche **Flusssteuerung** (*flow control*). Je nach Organisation der Flusssteuerung kann man verschiedene Übertragungsmodi unterscheiden.
Nennen Sie die drei Übertragungsmodi aus dem Kurstext.

- Store-and-forward-Modus
- Virtual-cut-through-Modus
- Wormhole-routing-Modus

4P

b) Gegeben sei ein Programm. Es benötigt auf einem Einprozessorsystem die Ausführungszeit T(1) = 1000. Dieses Programm wird auf ein Multiprozessorsystem portiert. Es kann entweder mit 10 oder mit 20 Prozessoren abgearbeitet werden und benötigt hierfür T(10) = 650 bzw. T(20) = 625. Berechnen Sie mit Hilfe von Amdahls Gesetz den prozentualen Anteil *a* des nur sequentiell ausführbaren Programmteils in beiden Fällen.
Hinweis: Für eine einfachere Rechnung geben Sie a bitte als Bruch an.

Amdahls Gesetz: $T(n) = (1-a)\frac{T(1)}{n} + a \cdot T(1) \Rightarrow a = \frac{n \cdot T(n) - T(1)}{n \cdot T(1) - T(1)}$

$$a(10) = \frac{10 \cdot 650 - 1000}{10 \cdot 1000 - 1000} = \frac{11}{18}$$

$$a(20) = \frac{20 \cdot 625 - 1000}{20 \cdot 1000 - 1000} = \frac{23}{38}$$

2P

c) Weshalb erhalten Sie unterschiedliche Werte für *a*?
Dies darf eigentlich nicht vorkommen. Ein Erklärung hierfür ist, dass das Amdahlsche Gesetz Synchronisations- und Kommunikationszeiten vernachlässigt.

3P

d) Berechnen Sie den Speedup S(10) und S(20). Berechnen Sie den maximal erreichbaren Speedup unter der Annahme a=0,6.
Hinweis: Für eine einfache Rechnung geben Sie Ihre Lösungen bitte als Bruch an.

$$S(n) = \frac{T(1)}{T(n)}$$

$$S(10) = \frac{T(1)}{T(10)} = \frac{1000}{650} = \frac{20}{13}$$

$$S(20) = \frac{T(1)}{T(10)} = \frac{1000}{625} = \frac{8}{5}$$

$$S_{\max} = \lim_{n\to\infty} \frac{T(1)}{\frac{(1-a)T(1)}{n} + aT(1)} = \frac{1}{a} = \frac{5}{3}$$

10P

e) Berechnen Sie die Effizienz E(10) und E(20). Wie viele Prozessoren können bei a=0,6 (a=3/5) maximal eingesetzt werden bevor die Effizienz unter 1/2 sinkt?

$$E(n) = \frac{S(n)}{n}$$

$$E(10) = \frac{S(10)}{10} = \frac{2}{13}$$

$$E(20) = \frac{S(20)}{20} = \frac{2}{25}$$

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{nT(n)} = \frac{T(1)}{T(1)(1-a) + naT(1)}$$

a eingesetzt:

$$E(n) = \frac{T(1)}{\frac{2}{5}T(1) + \frac{3}{5}nT(1)}$$

$$E(n) = \frac{1000}{400+600n}$$

E(n)=1/2 eingesetzt und aufgelöst nach n:

$$n = \frac{8}{3} = 2\frac{2}{3}$$

Es können also maximal 2 Prozessoren eingesetzt werden.

1P

f) Wie beurteilen Sie die Parallelisierbarkeit des Programms?
Die Parallelisierbarkeit des Programms ist sehr schlecht!

– ENDE –