Eingebettete Systeme WS 2005/2006

1 Aufgaben zu Kapitel 1

1. **Aufgabe:** (2 Punkte)
   Erklären Sie ausführlich, was man unter Eingebetteten Systemen versteht. Gehen Sie dabei auf ihre wesentlichen charakteristischen Eigenschaften ein. Präsentieren Sie mindestens drei Beispiele für eingebettete Systeme und begründen Sie, warum diese Ihrer Meinung nach zu den eingebettete Systemen gehören.

2. **Aufgabe:** (3 Punkte)
   Beschreiben Sie den Wachstum des Marktvolumens Eingebetteter Systeme. Recherchieren Sie dazu im Internet.

3. **Aufgabe:** (4 Punkte)
   Geben Sie eine formale Definition für die folgenden Eigenschaften von Eingebetten Systemen an:

   (a) Ausfallsicherheit - (reliability)

   (b) Wartbarkeit - (maintainability)

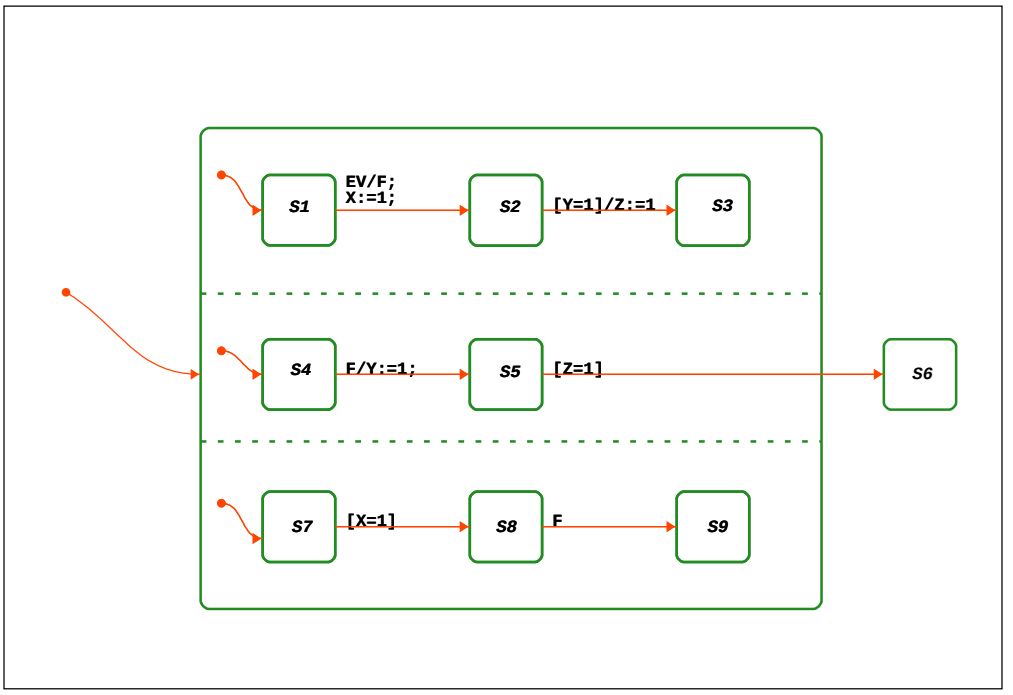
   (c) Erwartungswert für die Zeit bis zum Ausfall bzw. zur Instandsetzung - (MTTF, MTTR)

   (d) Verfügbarkeit - (availability)

4. **Aufgabe:** (1 Punkte)
   Sammeln Sie Informationen zu den Themen “ubiquitous computing” und “ambient intelligence” und stellen Sie diese angemessen dar. Benutzen Sie hierbei auch das Internet.

2 Aufgaben zu Kapitel 2

1. **Aufgabe:** (3 Punkte)
   Gegeben sei folgendes StateChart-Diagramm:



   Interpretieren Sie die Funktionsweise des StateCharts, wenn das Event EV im Zustand S1/S4/S7 mit X=0 auftritt.

   (a) Geben Sie die Events und Aktionen in den einzelnen Schritten an.

   (b) Geben Sie zu jedem Schritt die jeweilige Konfiguration an.

   (c) Begründen Sie, warum die Transition zwischen S8 und S9 nie schaltet. Wird bei dieser Eventfolge der Superzustand verlassen?

2. **Aufgabe:** (4 Punkte)
   Spezifizieren Sie eine Scheibenwischersteuerung. Der Autofahrer soll durch Antippen des Bedienhebels zwischen den Betriebsarten wählen können. Dabei soll folgendes Bedienkonzept umgesetzt werden:

   Sind die Scheibenwischer im ausgeschalteten Zustand, soll durch kurzes Antippen (kürzer als 1s) des Hebels ein einziger Wischvorgang ausgelöst werden. Wird der Hebel für eine längere Periode betätigt, wird in den Intervallmodus geschaltet.

   Im Intervallmodus wird ebenfalls durch kurzes Betätigen des Hebels ein zusätzlicher Wischvorgang durchgeführt, langes Betätigen schaltet in den Dauerbetrieb.

In dieser Betriebsart führt das kurze Betätigen des Hebels zum Beenden des Wischens. Langes Betätigen schaltet zurück in den Intervallmodus.

Zur Vereinfachung wird angenommen, dass ein Wischvorgang exakt eine Sekunde dauert, und die Scheibenwischer danach in der Ruheposition angelangt sind. Im Intervallbetrieb soll zwischen den Wischvorgängen eine Pause von 5 Sekunden eingelegt werden. Achten Sie bei der Modellierung darauf, dass die Scheibenwischer zwischen den Wischvorgängen immer in der Ruheposition abgelegt werden.

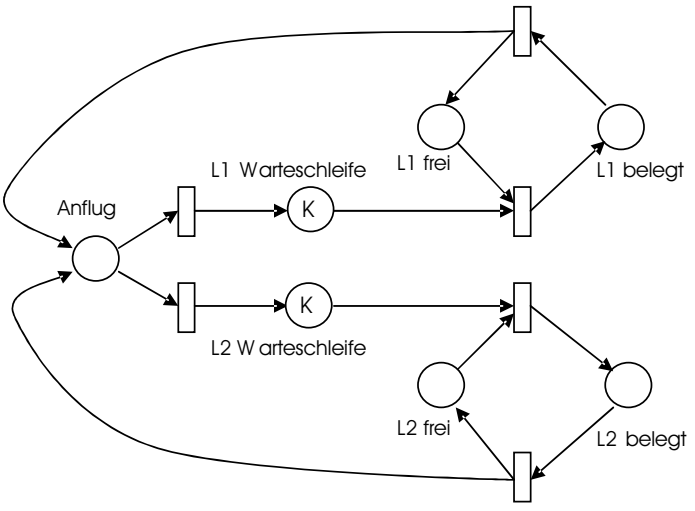
Entwerfen Sie das StateCharts Model.

3. **Aufgabe:** (3 Punkte)
   Überführen Sie das StateCharts Model eines Anrufbeantworters aus dem Buch (bzw. den Folien) in ein graphisches SDL Model.

4. **Aufgabe:** (4 Punkte)
   Simulieren Sie das StateChart-Modell für die Scheibenwischersteuerung von Übungsblatt 2 mit Statemate.

5. **Aufgabe:** (2 Punkte)
   Zeichnen Sie das Bedingungs-/Ereignis-System $N = (C, E, F)$, mit
   $C = \{c_1, c_2, c_3, c_4\}$,
   $E = \{e_1, e_2, e_3\}$,
   $F = \{(c_1, e_1), (c_1, e_2), (e_1, c_2), (e_1, c_3), (e_2, c_3), (e_2, c_4), (c_2, e_3), (c_3, e_3), (c_4, e_3), (e_3, c_1)\}$.
   Geben Sie den Vorbereich von e_3 sowie den Nachbereich von e_1 an. Ist N rein? Ist N schlicht? Begründen Sie Ihre Antworten.

6. **Aufgabe:** (4 Punkte)
   Betrachten Sie folgendes Petri-Netz:



Dieses Netz soll stark vereinfacht die Landebahnzuteilung eines mittelgroßen Flughafens modellieren. Der Fughafen hat zwei Landebahnen ausschließlich für ankommende Flugzeuge reserviert. Pro Landebahn wird eine Warteschlange verwaltet, in der maximal K Flugzeuge auf die Landeerlaubnis warten dürfen. Die Landebahnen sind so angeordnet, dass eine gleichzeitige Nutzung möglich ist. Wir nehmen an, dass sich N Flugzeuge im Anflug befinden.

(a) Vervollständigen Sie das S/T Netz. Geben Sie die Kantengewichte, die Kapazitäten der Stellen und eine sinnvolle Anfangsbelegung an.

(b) Wie müsste das Netz abgeändert werden, damit die maximalle Anzahl der Flugzeuge in der Warteschlange nicht in der Kapazität einer Stelle kodiert ist, sondern in der Anfangsbelegung mitkodiert werden kann?

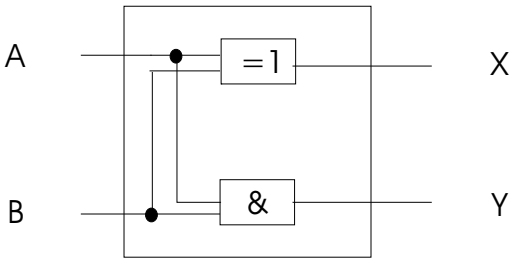
7. **Aufgabe:** (2 Punkte)
Erstellen Sie ein Weg/Zeit Diagramm, das einen Zyklus (z.B. Technologie-Zentrum > Eichlinghofen > Technologie-Zentrum) des H-Bahn Fahrplans im Taktbetrieb darstellt. Stellen Sie beide H-Bahn Linien in einem Diagramm dar. Die Zeitangaben können in Minuten - anfangend beim Zyklusbeginn $t = 0$ - erfolgen.

8. **Aufgabe:** (3 Punkte)
Spezifizieren Sie ein Kahn-Prozessnetzwerk, das die Folge der Fibonacci-Zahlen generiert. Verwenden Sie dabei atomare Prozesse wie die Addition zweier Zahlen, die Duplizierung einer Zahl sowie Initialisierungsprozesse, die anfänglich eine 1 generieren, und anschließend ihren Eingang weiterleiten. Die Ausgabe erfolgt über einen Prozess, der als Senke fungiert.

   Falls Sie das Kahn-Prozessnetzwerk in eine Hardware-Implementierung überführen müssten und davon ausgehen könnten, dass alle Prozesse parallel arbeiten und exakt die gleiche Ausführungszeiten haben, welche Aussagen über die Puffergrößen könnten Sie dann machen?

9. **Aufgabe:** (5 Punkte)

(a) Überlegen Sie sich, welchen häufig verwendeten Standard-Baustein die folgende Abbildung darstellt:



(b) Entwickeln Sie eine VHDL-Entity und einen Verhaltensrumpf jeweils für ein OR-, XOR-, NOR, und AND-Gatter mit zwei Eingängen. Nehmen Sie an, daß die Gatterverzögerung einheitlich 2ns beträgt.

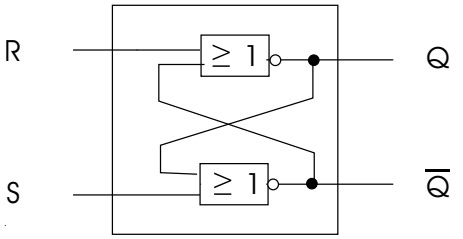
(c) Erstellen Sie eine Strukturbeschreibung in VHDL für die in der Abbildung oben dargestellten Schaltung. Verwenden Sie dabei nur die Gattertypen aus Aufgabenteil (b).

10. **Aufgabe:** (5 Punkte)
In der Vorlesung wurde ein Volladdierer in der Hardwarebeschreibungssprache VHDL entworfen. Überlegen Sie, wie man diesen Volladdierer zu einem N-Bit Addierer mit 'ripple carry'-Berechnung verallgemeinern kann:

    (a) (3 Punkte) Zeichnen Sie die allgemeine Struktur eines N-Bit Addierers unter der Verwendung von Volladdierern. Nutzen Sie dabei die Regularität der Verbindungen der 'inneren' Volladdierer aus.

    (b) (2 Punkte) Geben Sie für N=4 Bit eine strukturelle Beschreibung eines N-Bit Addierers in VHDL an. Verwenden Sie als Bausteine die in der Vorlesung vorgestellten Volladdierer.

11. **Aufgabe:** (5 Punkte)
Ein RS-Latch ist ein bistabiles Speicherelement, das aus zwei rückgekoppelten NOR-Bausteinen aufgebaut ist:



Geben Sie eine Verhaltensbeschreibung eines RS-Latch in VHDL an. Nehmen Sie an, daß die verwendeten NOR-Bausteine eine Verzögerung von je 1 ns haben. Verwenden Sie zur Darstellung aller Signale den Typ BIT. Zeichnen Sie die Signalverläufe, wenn Ihr RS-Latch mit folgenden Input-Daten (Stimuli) versehen wird:

S <= '0' after 0 ns, '1' after 5 ns, '0' after 10 ns
R <= '0' after 0 ns, '1' after 15 ns, '0' after 20 ns

Berücksichtigen Sie bei der Darstellung auch die internen Signale des Latchs und geben Sie die Werte für alle Deltazklen an! Was ist bei der Initialisierung der Signale zu beachten?

3 Aufgaben zu Kapitel 3

1. **Aufgabe:** (4 Punkte)
Versehen Sie Ihren Roboter mit einer Stoßstange, die einen Drucksensor enthält und so Zusammenstöße mit anderen Objekten an die RCX-Einheit melden kann. Überprüfen Sie die korrekte Funktionsweise des Sensors mit Hilfe des LC-Displays der RCX-Einheit. Vorher sollte der Sensor durch ein kurzes `nqc`-Programm als `SENSOR_TOUCH` konfiguriert werden.

    - Schreiben Sie ein `nqc`-Programm, das es dem Roboter ermöglicht, Hindernissen, mit denen er beim Herumfahren kollidiert ist, auszuweichen. Übertragen Sie das Programm auf die RCX-Einheit des Roboters und überprüfen Sie das Verhalten des Roboters. Wie kann erreicht werden, daß der Roboter mit einem Objekt nicht mehrfach direkt hintereinander kollidiert? Wie verhält sich Ihr Roboter, nachdem er in eine Zimmerecke gefahren ist?

2. **Aufgabe:** (6 Punkte)
   Versehen Sie Ihren Roverbot aus der ersten Aufgabe mit einem nach unten gerichteten Lichtsensor (s. Konstruktionsanleitung). Mit dessen Hilfe kann der Roboter dazu gebracht werden, auf einem schwarzen Strich (der ein Oval bildet) entlangzufahren.

   - Stellen Sie zunächst fest, in welchem Bereich die Helligkeitswerte des Sensors schwanken, wenn sich Ihr Roboter über eine helle oder über eine dunkle Fläche bewegt (diese Werte sind vom Umgebungslicht abhängig). Konfigurieren Sie Ihre RCX-Einheit so, dass die Information des Lichtsensors geeignet abgefragt werden kann. Die ermittelten Werte sollten so in das `nqc`-Programm eingebaut werden, dass sich das Programm leicht an andere Lichtverhältnisse anpassen läßt.
   - Programmieren Sie Ihren Roboter zunächst mit einem möglichst kurzen Programm so, dass er von einem beliebigen Startpunkt aus die schwarze Linie findet und auf dieser entlangfährt. Legen Sie dieses Programm auf Programmspeicherplatz 1 ab.
   - Verfeinern Sie Ihr Programm so, dass der Roboter möglichst effizient auf der Linie entlangfährt (z.B. keine Richtungswechsel auf der Linie oder kein langes Herumirren, wenn die Linie verlassen wurde). Speichern Sie dieses Programm auf Programmspeicherplatz 2 ab.

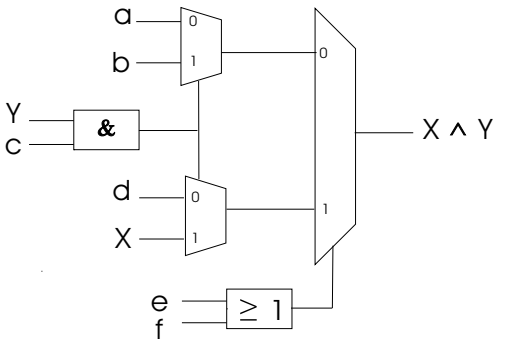
3. **Aufgabe:** (4 Punkte)
   Zur Umwandlung analoger Signale in digitale Werte wurde in der Vorlesung u.a. die Wandlung nach dem Wägeprinzip (successive approximation) vorgestellt.
   Führen Sie unter Anwendung des Wägeprinzips eine Wandlung der Eingangsspannungswerte U_{in} = 2,25 V, 3,75 V und 1,8 V in die entsprechende Binärdarstellung durch, indem Sie jeweils

   - die zu einem Zeitpunkt angelegte Vergleichsspanung U_{ref} und
   - die nach einem Vergleich feststehende Binärzahl

   angeben.
   Die Digitaldarstellung soll mit einer Genauigkeit von n=4 Bit erfolgen. Es wird des weiteren angenommen, daß der Arbeitsbereich des A/D-Wandlers zwischen U_{min} = 1V (0000_2) und U_{max} = 4,75V (1111_2) liegt.

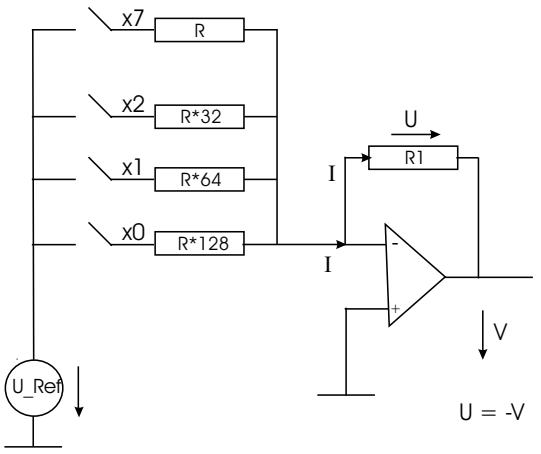
4. **Aufgabe:** (3 Punkte)
   Zur Realisierung von Eingebetteten Systemen werden in zunehmenden Maße Prozessoren eingesetzt. In der Vorlesung wurden Mikrocontroller, DSP- und VLIW-Prozessoren vorgestellt.
   Stellen Sie die o.g. Prozessortypen bezüglich Einsatzgebiet, Anforderungen und Spezifika des Befehlssatzes und des Speichers gegenüber.

5. **Aufgabe:** (3 Punkte)
Gegeben sei folgender CLB aus einem FPGA der Firma ACTEL:



- Geben Sie für den angegebenen Multiplexer-basierten CLB eine Belegung der Eingänge a, b, c, d, e und f an, so daß die Funktion $x \wedge y$ realisiert wird.
- Realisieren Sie die beiden Booleschen Funktionen $x \vee y$ bzw. $\neg x$ mit Hilfe eines Multiplexers.

6. **Aufgabe:** (3 Punkte)
Eine Möglichkeit zur D/A-Wandlung besteht in der Verwendung eines Stromsummenwandlers (s. Abbildung).



Gegeben seien im folgenden: $U_{ref} = 5V, R = 1k\Omega$ und n = 8 bit. Des weiteren wird die Existenz eines idealen Operationsverstärkers angenommen.

- Wie groß ist der Rückkopplungswiderstand R_1 zu wählen, wenn die Digitalwerte in einen Spannungsbereich von $U_{min} = 0$V und $U_{max} = 10$V abgebildet werden soll?
- Die Genauigkeit der D/A-Wandlung mittels Stromsummenwandler ist stark abhängig von der Genauigkeit (Toleranz) der verwendeten Widerstände. Bei zu großen Abweichungen der Widerstände vom Sollwert kann es vorkommen, daß der für einen bestimmten Dualwert (d) gelieferte Strom geringer ist, als der für den nächst geringeren Dualwert(d-1) (Monotoniefehler).
Die größte Auswirkung eines solchen Monotoniefehlers ist bei den Werten 01111111_2 und 10000000_2 zu erwarten.

- Berechnen Sie für die Werte 01111111_2 und 10000000_2 jeweils den Stromfluß.
- Um wieviel Ω darf der Wert von Widerstand R maximal nach oben abweichen, damit kein Monotoniefehler auftritt?

4 Aufgaben zu Kapitel 4

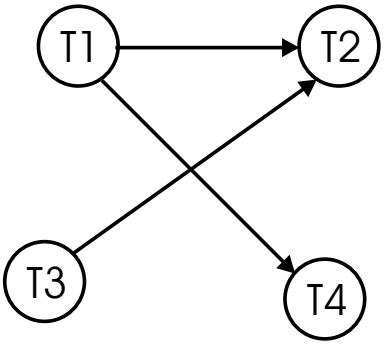
1. **Aufgabe:** (3 Punkte)
   Was ist der Unterschied zwischen statischem und dynamischem Scheduling? Nennen sind die Vor- und Nachteile von statischem und dynamischem Scheduling.
   Was versteht man unter einem preemptiven bzw. einem nicht-preemptiven Scheduling-Verfahren?

2. **Aufgabe:** (4 Punkte)
   Betrachten Sie folgende Taskmenge, wobei A_n die Ankunftszeit, d_n die Deadline und c_n die Ausführungszeit darstellt:

   $T_1 : A_1 = 7, d_1 = 16, c_1 = 4$
   $T_2 : A_2 = 4, d_2 = 15, c_1 = 5$
   $T_3 : A_3 = 11, d_3 = 20, c_1 = 3$
   $T_4 : A_4 = 0, d_4 = 26, c_1 = 10$

   Erstellen Sie für diese Taskmenge je einen Schedule nach den Scheduling-Verfahren EDF und LL. Zeigen Sie in Form eines Diagramms, welche Task zu welchem Zeitpunkt aktiv sind. Verpaßt ein Task seine Deadline?

3. **Aufgabe:** (3 Punkte)
   Gegeben seien die folgenden Prozesse T_1 bis T_4. Die Abhängigkeiten der Tasks untereinander werden durch den dargestellten Taskgraphen beschrieben. c_i bezeichnet die Ausführungszeit und d_i das Deadline-Intervall des jeweiligen Task.



- T_1 mit $c_1 = 2$ und $d_1 = 10$,
- T_2 mit $c_2 = 5$ und $d_2 = 20$,
- T_3 mit $c_3 = 4$ und $d_3 = 12$,
- T_4 mit $c_3 = 2$ und $d_3 = 15$

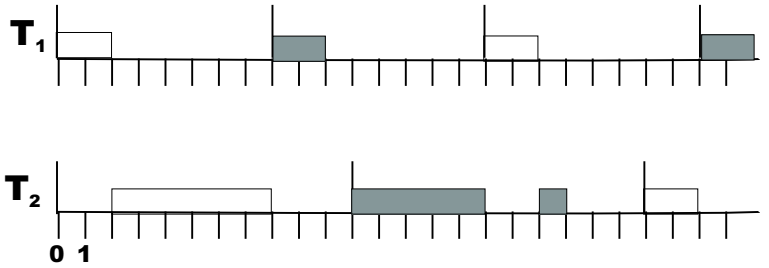
Führen Sie für die gegebene Taskmenge ein Scheduling mittels des *Latest Deadline First* Algorithmus durch.

4. **Aufgabe:** (4 Punkte)
   Gegeben seien die folgenden voneinander unabhängigen Prozesse:

   - T_1 mit $c_1 = 1$ und $d_1 = p_1 = 4$,
   - T_2 mit $c_2 = 2$ und $d_2 = p_2 = 5$,
   - T_3 mit $c_3 = 2$ und $d_3 = p_3 = 8$

   p_i stellt die Periode des jeweiligen Task T_i dar. Führen Sie für ein Einprozessorsystem ein Scheduling mittels *Rate Monotonic Scheduling* durch. Kann ein Task seine Deadline verpassen?

5. **Aufgabe:** (3 Punkte)
   Gegeben ist folgendes Taskdiagramm:



   Geben Sie die Parameter der Tasks an (c_n, $p_n = d_n$). Mit welchem Schedulingverfahren könnten die Tasks geschedult worden sein? Zeichnen Sie für ein weiteres Schedulingverfahren ein Taskdiagramm dieser Tasks (z.B. RMS oder EDF).

6. **Aufgabe:** (3 Punkte)
   Betrachten Sie folgende Taskmenge, wobei A_n die Ankunftszeit darstellt und c_n die Ausführungszeit. $\Delta t(P(S))$ gibt an, in welchem Taktzyklus -relativ zu A_n- der Task die Ressource S anfordert. Ensprechend kennzeichnet $\Delta t(V(S))$ die Freigabe:
   $T_1 : A_1 = 7, c_1 = 8, \Delta t(P(S)) = 4, \Delta t(V(S)) = 6$
   $T_2 : A_2 = 4, c_1 = 9, \Delta t(P(S)) = 1, \Delta t(V(S)) = 6$
   $T_3 : A_3 = 11, c_1 = 3, \Delta t(P(S)) = -, \Delta t(V(S)) = -$
   $T_4 : A_4 = 0, c_1 = 10, \Delta t(P(S)) = 3, \Delta t(V(S)) = 8$
   Die Prioritäten sind wie folgt vergeben: $T_1 > T_2 > T_3 > T_4$. Die Tasks werden auf einer CPU mit einem preemptiven Scheduler nach ihren statischen Prioritäten geschedult.

   (a) Es wird zunächst keine Prioritätsvererbung eingesetzt. Stellen Sie in einem Diagramm dar, welcher Task zu welchem Zeitpunkt aktiv ist. Kennzeichnen Sie die Zeitabschnitte, in denen es zu Prioritätsumkehr kommt. Geben Sie dabei für jeden Abschnitt an, welche Tasks durch welche anderen Tasks blockiert werden.

   (b) Welcher Schedule ergibt sich, wenn das *priority inheritance*-Protokoll benutzt wird? Stellen Sie den Schedule in einem Diagramm dar.

7. **Aufgabe:** (3 Punkte)
Der Bau eines (kleinen) Hauses soll mit Hilfe eines Task-Graphen geplant und geschedulet werden. Gegeben sind folgende Teilaufgaben (Tasks), die die in der Tabelle (siehe Rückseite) angegebene Zeit in Tagen dauern und die sinnvollerweise erst dann angefangen werden können, wenn ihre jeweiligen Vorgänger abgeschlossen sind.

Task	Vorgang	Zeit	Vorgänger
A	Angebot einholen	7	
B	Finanzierung planen	2	
C	Fundament	3	A, B
D	Außenwände mauern	10	C
E	Fenster einsetzen	1	D
F	Außenwände verputzen	3	E
G	Dachstuhl bauen	4	D
H	Dach decken	2	G
I	Fertigstellung	1	F, H

Stellen Sie gemäß der Tabelle einen Task-Graphen auf. Geben Sie für jeden Task Start- und Endzeit an, wenn die Tasks nach der

(a) ASAP-Regel (As Soon As Possible - jeder Task beginnt frühestmöglich)

(b) ALAP-Regel (As Late As Possible - jeder Task beginnt spätestmöglich)

geschedulet werden. Bei Verwendung der ALAP-Regel soll das Haus in spätestens 30 Tagen komplett fertig sein.

5 Aufgaben zu Kapitel 5

1. **Aufgabe:** (2 Punkte)
Beantworten Sie folgende Fragen:

   (a) Welche Unterschiede gibt es zwischen RTOS und Standard OS?

   (b) Welche Nachteile kann der Einsatz von RTOS mit sich bringen?

   (c) Warum kann auf die Abschottung der Hardware gegenüber der Anwendung verzichtet werden?

   (d) Welche Ansätze gibt es RTOS-ähnliche Eigenschaften in Standard OS einzubauen?

2. **Aufgabe:** (2 Punkte)
Beschreiben Sie den Unterschied zwischen einer internen und einer externen Synchronisation von Uhren. Welche Probleme treten jeweils auf?

3. **Aufgabe:** (5 Punkte)
   Optimieren Sie das nachfolgende Programm:

```
1  #include <stdio.h>
2
3  #define DATALEN    15
4  #define FILTERTAPS 5
5
6  double x[DATALEN]    = { 128.0, 130.0, 180.0, 140.0, 120.0,
7                           110.0, 107.0, 103.5, 102.0,  90.0,
8                            84.0,  70.0,  30.0,  77.3,  95.7  };
9
10 const double h[FILTERTAPS] = { 0.125, -0.25, 0.5, -0.25, 0.125 };
11
12 double y[DATALEN];   // result;
13
14 int main()
15 {
16   int i,n;
17
18   for(i=0;i<DATALEN;++i)
19   {
20     y[i]=0;
21     for(n=0;n<FILTERTAPS;++n)
22       if ((i-n)>=0) y[i]+=h[n]*x[i-n];
23   }
24
25   for(i=0;i<DATALEN;++i) printf("%.2f ",y[i]);
26   return 0;
27 }
```

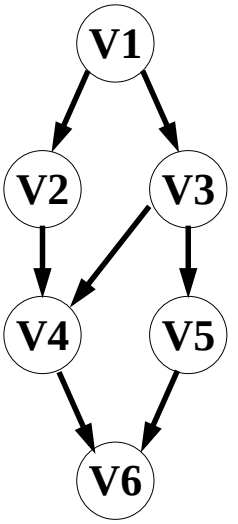
   Führen Sie mindestens folgende Optimierungen durch:

   - Entfernen des `if` aus der inneren Schleife (Zeile 22).
   - Loopunrolling (Zeile 21).
   - Float to fixpoint conversion.

   Geben Sie für jeden Optimierungsschritt die optimierte Version Ihres Programms und die durch das Programm erzeugte Ausgabe ab.
   Hinweis: Der Quelltext ist über die Übungsgruppenseite abrufbar. Sie können den `gcc` in Ihrem embsXXX Account zum Compilieren verwenden.

4. **Aufgabe:** (5 Punkte)

Gegeben sei ein Graph mit den Knoten $v_1, \ldots, v_6$ (s. Abb. rechts), die jeweils einzelne Code-Segmente in VHDL repräsentieren. Ferner sei eine Zielarchitektur bestehend aus einem Prozessor und einem FPGA (Field Programmable Gate Array) gegeben, auf der das System realisiert werden soll. Die Realisierungskosten für das FPGA in Form von CLBs (configurable logic blocks) sowie die Ausführungszeiten für eine mögliche Software- bzw. Hardware-Realisierung von $v_1, \ldots, v_6$ sind nachfolgend angegeben:



	v_1	v_2	v_3	v_4	v_5	v_6
#CLBs	50	200	170	150	80	10
Zeit_{HW}	8	3	15	7	4	2
Zeit_{SW}	30	10	25	15	18	5

Versuchen Sie für alle Knoten v_i eine Zuordnung zu Software bzw. Hardware mit minimalen Kosten zu finden, wobei die Gesamtausführungszeit des Systems aufgrund von Echtzeitbedingungen nicht größer als 65 Zeiteinheiten werden darf. Reine Software- oder Hardwarelösungen scheiden aufgrund der zu langen Ausführungszeiten (103 Zeiteinheiten mit 0 CLBs) oder zu hohen Kosten (660 CLBs bei 32 Zeiteinheiten) von vornherein aus. Es gilt hier also einen Kompromiss von Kosten und Gesamtausführungszeit unter Berücksichtigung der gegebenen Zeitschranke zu finden.

5. **Aufgabe:** (3 Punkte)

In der Vorlesung wurde ein IP-Modell für die optimale Verwendung von mehreren partitionierten Scratchpad-Speichern vorgestellt. Dieser Ansatz läßt sich auch dazu verwenden, die in aktuellen ARM-Prozessoren verfügbaren "Tightly Coupled Memories" (TCMs) auszunutzen. Allerdings verfügen diese Prozessoren über eine Harvard-Architektur, d.h. separate Speicher und Busse für Daten und Instruktionen. Folglich hat ein solcher Prozessor zwei TCMs: eines ausschließlich für Daten, eines für Instruktionen.

Überlegen Sie, wie man das vorgestellte Modell für partitionierte Speicher erweitern kann, so dass eine Allokation auf zwei TCMs in einer Harvard-Architektur realisiert wird. Wie sieht das Gleichungssystem im allgemeinen Fall, also für k Daten- und l Instruktions-TCMs aus?

6. **Aufgabe:** (3 Punkte)

Die Optimierungstechnik *Register Pipelining* kann in optimierenden Compilern für eingebettete System eingesetzt werden, um redundante Speicherlesezugriffe zu eliminieren. Hierzu wird vor der zu optimierenden inneren Schleife ein Prolog generiert, der die korrekte Ausführung der ersten Iteration garantiert. Am Ende der Schleife wird eine Reihe von Register-Kopieroperationen erzeugt, die die sogenannte *Register Pipeline* darstellen.

Optimieren Sie folgende Schleife (in Pseudo-C-Syntax), so daß pro Iteration nur noch ein Speicherzugriff erfolgt:

```
for( i = 0; i < 100; i++ )
```

```
{

  R1 = a[i];
  R2 = a[i + 2];
  R3 += R1 + R2;

}
```

7. **Aufgabe:** (4 Punkte)
   Angenommen Sie haben ein Rechnersystem, das neben dem Hauptspeicher einen Scratchpad-Speicher enthält.

Speicher	Größe in Bytes	E/Zugriff
Scratchpad	2k	1.1 nJ
Hauptspeicher	256k	31 nJ

Variable	Größe in Bytes	Zugriffe
a	4	16
b	512	2048
c	1024	1024
i	4	8192
j	4	12
x	256	1024
y	256	512

   In der linken Tabelle ist der Energiebedarf pro Zugriff dargestellt. Eine weitere Tabelle listet die Variablen des auszuführenden Programms auf.

   Welche Variablen sollten im Scratchpad-Speicher abgelegt werden, damit das Programm möglichst energieeffizient ausgeführt werden kann?

   Lösen Sie das Problem mit Hilfe eines IP-Modells. Geben Sie sowohl das IP-Modell als auch die ermittelte Variablenauswahl an. Sie können das IP-Modell auf die von Ihnen bevorzugte Weise lösen. Es ist nicht notwendig die einzelnen Lösungsschritte anzugeben. **Hinweis:** Testversion eines Solvers gibt es bei der Firma LINDO: *http://www.lindo.com*

8. **Aufgabe:** (3 Punkte)
   Betrachten Sie für die Variablen {a, b, c, d, e, f} folgende Zugriffssequenz:

   e c f b e a b e a d e d b e b f b

   Die Rechnerarchitektur, die diese Sequenz ausführt, hat folgende Kenndaten:

   - Ein Adressregister (AR).
   - Zugriffe auf den Speicher beziehen die Adresse immer aus dem AR.
   - Inkrementieren/Dekrementieren des AR um Eins kann im Lade- bzw. Speicherbefehl mitkodiert werden (Postinkrement/-dekrement).
   - Alle weiteren Änderungen des AR (Setzen, +/- Offset) bedürfen eines separaten Befehls.

   Ermitteln Sie eine optimale Variablenanordnung unter dem Gesichtspunkt möglichst weniger expliziter Adressberechnungen.

   Schreiben Sie ein Assemblerprogramm, dass diese Zugriffssequenz auf ihre Variablenanordnung erzeugt. Nehmen Sie dazu an, dass alle Zugriffe lesend sind. Verwenden Sie eine übliche Assemblersyntax:

`LD r,(AR), LD r,(AR)++, LD r,(AR)--,`
`LI AR,#const, ADDI AR,#const, SUBI AR,#const`

9. **Aufgabe:** (3 Punkte)
Ermitteln Sie für die untenstehende Zugriffssequenz der Variablen {v, w, x, y, z,} eine optimale Speicherbelegung mit möglichst wenigen Adressberechnungen (Kenndaten siehe Aufgabe 1). Geben Sie neben der Speicherbelegung auch die zugehörigen Interferenz- und Zugriffsgraphen an.

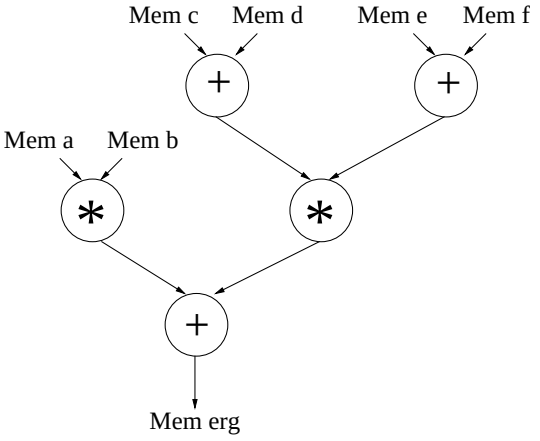
x w y x y v w z x w z x z w v w x z

10. **Aufgabe:** (4 Punkte)
Gegeben seien ein Datenflussgraph und eine Auswahl von Prozessor-Instruktionen. Die im Datenflussgraphen verwendete Notation "Mem X" soll ausdrücken, dass eine Variable im Speicher vorliegt, bzw. in den Speicher geschrieben werden soll. In geschweiften Klammern stehende Register (r_1, r_2 und r_3) der Prozessor-Instruktionen können alternativ verwendet werden. So sind z.B. in Zeile I_1 die Prozessor-Instruktionen $r_1 = Mem$ und $r_2 = Mem$ erlaubt.

   (a) Führen Sie für den dargestellten Datenflussgraph eine Überdeckung mit Prozessor-Instruktionen (*Codeselektion*) durch, so dass die Kosten minimiert werden.

   (b) Bestimmen Sie für die ausgewählten Instruktionen eine Ausführungsreihenfolge (*Instruktionsscheduling*), bei der die wenigsten *Spills* (Retten von Registerinhalten in den Speicher) erforderlich sind und fügen Sie den dazu erforderlichen *Spillcode* (zusätzliche Instruktionen zum Speichern und Laden der geretteten Werte) ein. Welche Kosten entstehen nun?

   (c) Gibt es unter Berücksichtigung von Spillcode eine bessere Überdeckung des Datenflussgraphen?



	Prozessor-Instruktionen	Kosten
I_1	$\{r_1, r_2\} = Mem$	2
I_2	$Mem = \{r_1, r_2, r_3\}$	2
I_3	$r_3 = \{r_1, r_2\}$	1
I_4	$\{r_1, r_2\} = r_3$	1
I_5	$\{r_2, r_3\} = r_1 + r_2$	2
I_6	$\{r_1, r_3\} = r_1 * r_2$	2
I_7	$r_1 = r_3 * r_2 + r_1$	3