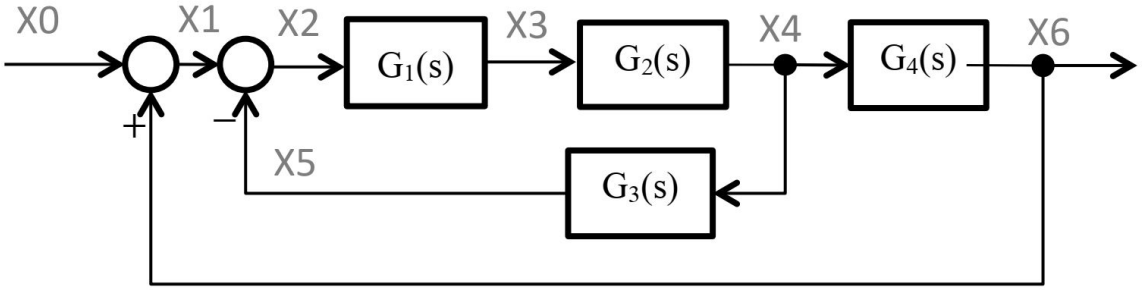
Klausur SS 2017

Aufgabe 1 Regelung

Teilaufgabe 1 Zeitkontinuierliche Regelung

a) Regelung: mit Rückführung, Steuerung: ohne Rückführung
b) Laplace-Übertragungsfunktion



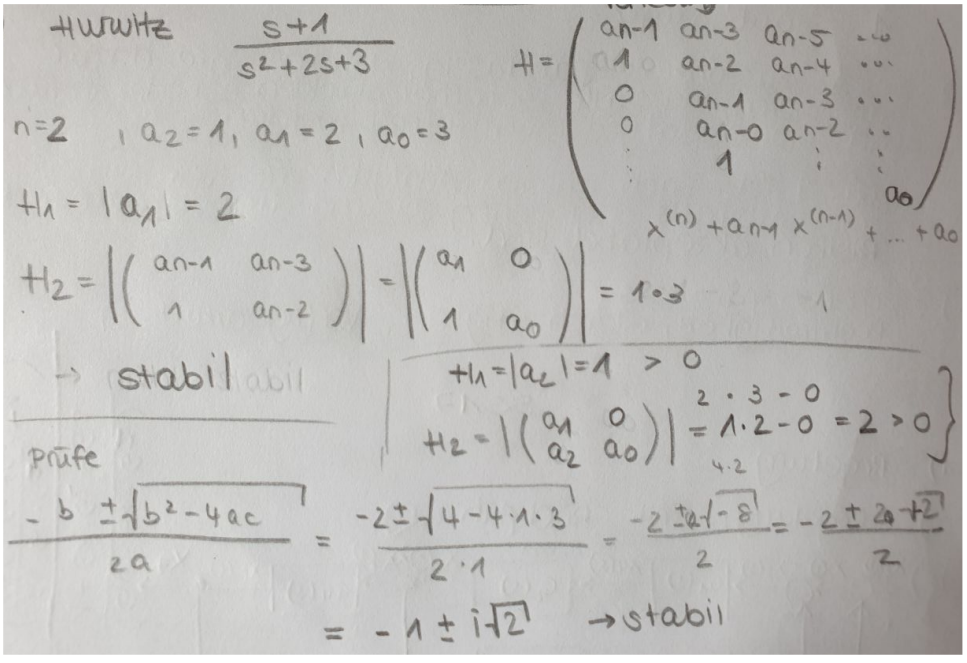
$x_1 = x_0 + x_6$, $x_6 = G_4 \cdot x_4$
Rückkopplung (Gegenkopplung): $x_4 = (G_2 \cdot G_1)/(1 + G_2 \cdot G_1 \cdot G_3) \cdot x_1$

$x_6 = G_4 \cdot [(G_2 \cdot G_1)/(1 + G_2 \cdot G_1 \cdot G_3) \cdot x_1] = G_4 \cdot [(G_2 \cdot G_1)/(1 + G_2 \cdot G_1 \cdot G_3) \cdot (x_0 + x_6)]$
$= [(G_4 \cdot G_2 \cdot G_1)/(1 + G_2 \cdot G_1 \cdot G_3) \cdot x_0] + [(G_4 \cdot G_2 \cdot G_1)/(1 + G_2 \cdot G_1 \cdot G_3) \cdot x_6]$
$\Leftrightarrow -(G_4 \cdot G_2 \cdot G_1)/(1 + G_2 \cdot G_1 \cdot G_3) \cdot x_0 = [(G_4 \cdot G_2 \cdot G_1)/(1 + G_2 \cdot G_1 \cdot G_3) - 1] \cdot x_6$
$\Leftrightarrow x_6 = [(G_4 \cdot G_2 \cdot G_1)/(1 + G_2 \cdot G_1 \cdot G_3) \cdot x_0]/[1 - (G_4 \cdot G_2 \cdot G_1)/(1 + G_2 \cdot G_1 \cdot G_3)]$

Geht wahrscheinlich einfacher, wenn man die Formel für die Mitkopplung verwendet für x6 (ist ja auch das gleiche wie Rückkopplung wenn man -G_3 statt G_3 macht)

c) $2[s(sX(s) - x(0)) - x(0)] - 7(s\,X(s) - x(0)) + 2X(s) = -2W(s)$
$\Leftrightarrow 2s^2 X(s) - 7sX(s) + 2X(s) = -2W(s) \Rightarrow G(s) = X(s)/W(s) = -2/(2s^2 - 7s + 2)$

d) Hurwitz



e) $$R(s) = K_p\left(1 + \frac{1}{T_N s} + T_v s\right) = K_p \frac{T_V T_N s^2 + T_N s + 1}{T_N s}$$

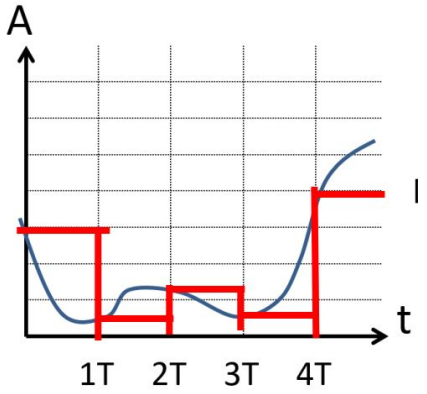
Teilaufgabe 2 Zeitdiskrete Regelung

a) $f_{max} < 20kHz \Rightarrow T = 1/(2f_{max}) = 2.5 \cdot 10^{-5} s$ maximaler Abtastperiode

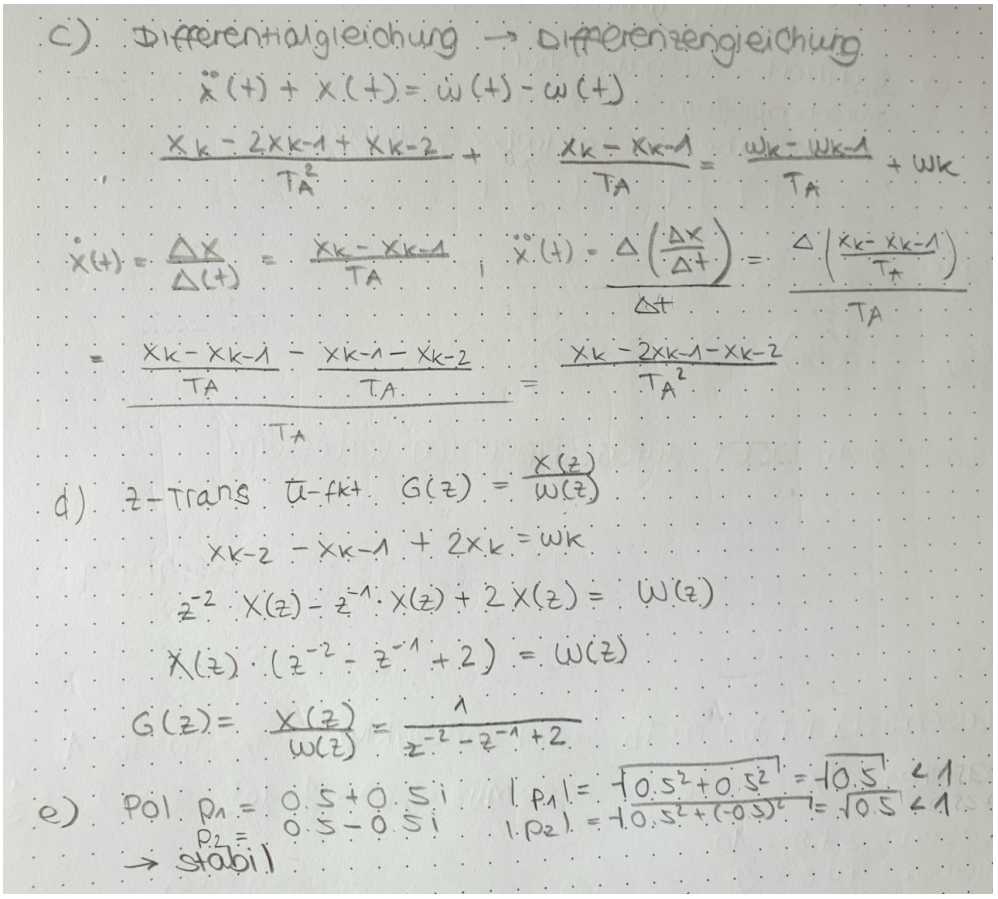
Sollte das nicht 1/40kHz = 1/40 ms sein?

Wenn man mit zu kleinen Abtastfrequenz (zu großen Abtastperiode) abtastet, kann es zu Aliasing-Effekten kommen. Beim Abtasten werden hoherfrequente Anteile “verpasst”, sodass das resultierende Signal sich als ein niederfrequenteres Signal “ausgibt” als das Originalsignal.

b) Sample and Hold

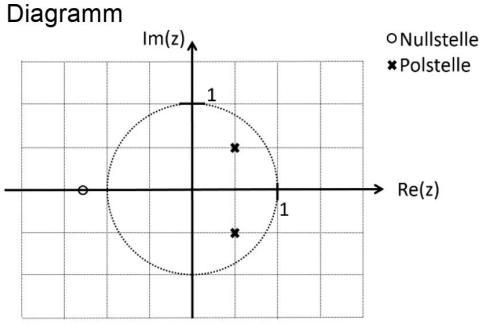


c), d), e)



f) Zusätzliche Informationen aus diesem Pol-Nullstellen-Diagramm?

Pol-Nullstellen Diagramm



Aufgabe 2 Rechnerarchitekturen, Busse, Operationsverstärker, Analog-/Digitaltechnik

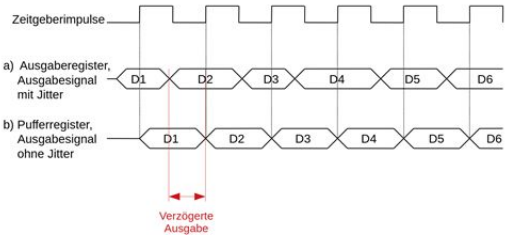
Teilaufgabe 1 Rechnerarchitekturen und Busse

a) Startwert $1000 = 10^3$ Takt $2Mhz = 2 \cdot 10^6\ 1/s$
Reset nach $10^3/(2 \cdot 10^6\ 1/s\ ) = 5 \cdot 10^{-4} s$

Beispielapplikation
- Mars Rover, entfernte eingebette Systeme, die nicht physisch von Menschen erreicht werden können
- Textverarbeitungsprogramm muss Rückmeldung geben

b) Je nachdem, auf welchen Speicher zugegriffen wird, variiert die Zugriffszeit (Schnell, teuer bis langsam, billig)
→ Zeitliche Fluktuation in der Ausführungszeit sichtbar

Gegenmaßnahme: **Echtzeit-Ausgabeeinheit**



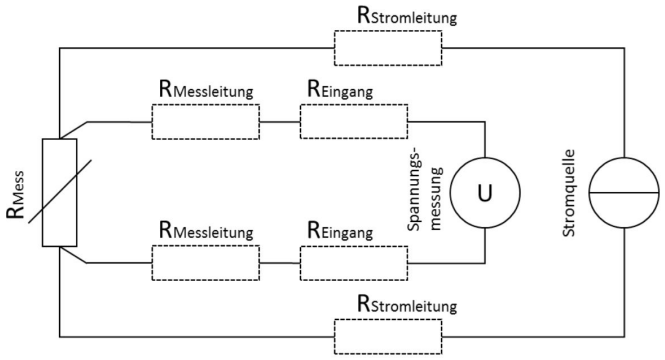
- Grundgedanke: **Zwischenpuffern** der Ausgabewerte
- Ports, deren Aus- und Eingabezeitverhalte von einem **Zeitgeber gesteuert**, **nicht von der Software**

c) Nein, da im worst case die Pipeline zurückgesetzt werden muss (zb unvorhergesagter Sprung)

d) Lesen: 2+n=10 (-> Wartezyklus), Schreiben: 1+n=9
Wie lange braucht PCI-Master eines 66Mhz Busses für Zugriff?
Lesen: $10/66 \cdot 10^{-3}\ ms$ Schreiben: $9/66 \cdot 10^{-3}\ ms$

e) 1. Flexibler, man kann neue Sensoren hinzufügen und entfernen ohne etwas zu ändern 2. (?) Alle haben die gleiche Schnittstelle -> einfacher

Teilaufgabe 2 Operationsverstärker und Analog-/Digitaltechnik



a) normaler nichtinvertierender Verstärker? Oder Subtrahierer nach Klausur WS17/18 Aufgabe 2.2

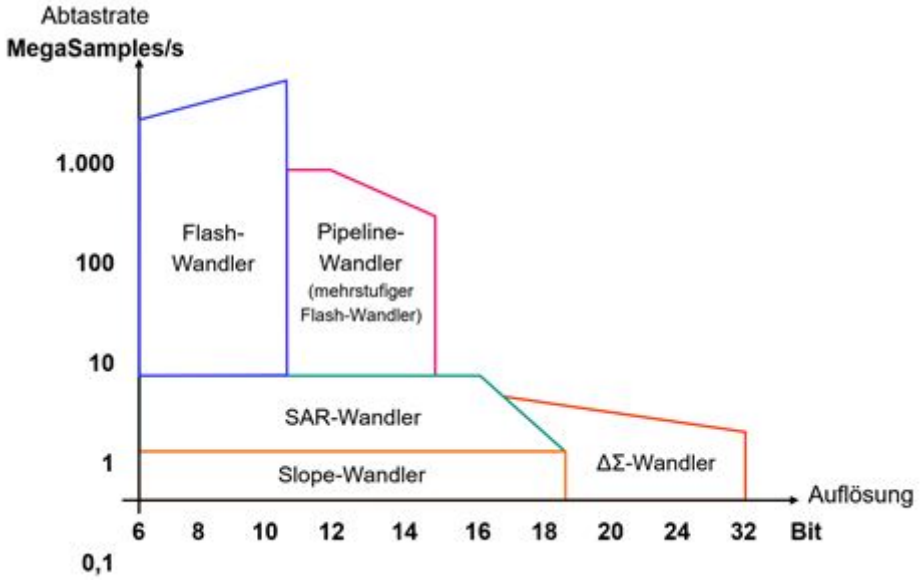
b) Höchste Auflösung -> Delta-Sigma-Wandler

c) SPS -> nicht relevant

Aufgabe 3 Echtzeitkommunikation und Echtzeitprogrammierung

Teilaufgabe 1 Echtzeitkommunikation

a) 00 111 01010 alles invertiert da erstes bit 1: 11 000 10101



b) Zeichne NRZ: High 1, Low: 0

c) Taktsignal

d) ?: + Jitter verbessern, Potentiell kann man in batches senden=weniger Overhead, - Zeitliche Verzögerung (Latenz)

e) I4 darf senden

f) (4,3,4, /, 2, 4, 2)

Teilaufgabe 2 Echtzeitscheduling

a) Logische Korrektheit

b) Zeitliche Korrektheit

c) Wahrung der Rechtzeitigkeit (Einhalten von Deadlines), Gleichzeitigkeit (Einhalten der Rechtzeitigkeit von mehreren Aktionen), Verfügbarkeit

d) Anzahl Aufgaben m, Anzahl Prozessoren n

Echt parallel $m = n$

Für jede Aufgabe steht ein Prozessor zur Verfügung. Alle können parallel abgearbeitet werden.

Quasi parallel $m > n$

Spezialfall: $n = 1$ Einprozessorsysteme

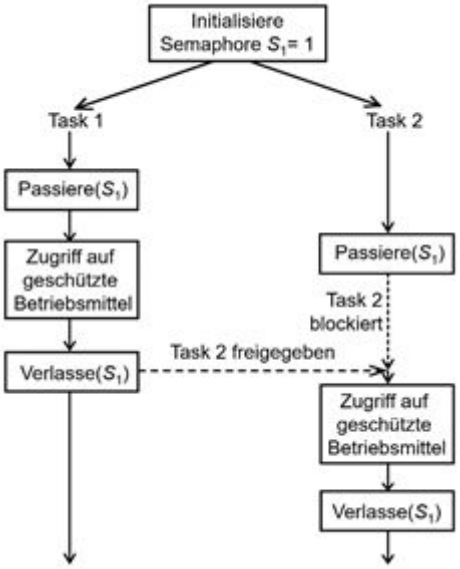
Es stehen weniger Prozessoren zur Verfügung als es Aufgaben gibt. Man benötigt eine Echtzeitschedulingsstrategie, um die Wahrung der Gleichzeitigkeit zu gewährleisten, die einen Schedule findet, sofern er existiert.

e) Laxity $d_i - (t + er_i)$: Kombination aus Deadline minus dem aktuellen Zeitschritt und der restlichen Ausführungszeit eines Tasks i ("wie viel spielraum hätte man noch wenn man den task *jetzt* zuende laufen lassen würde")
Dem Task mit der kleinsten Laxity zum Zeitpunkt t, wird der Prozessor zugeteilt.
-> Dynamische Prioritäten

Aufgabe 4 Echtzeitbetriebssysteme und SPS

Teilaufgabe 1 Echtzeitbetriebssysteme

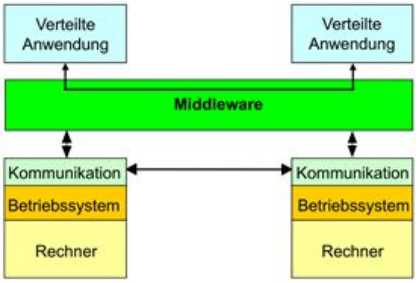
a)



b) Logische Adresse: 32 Bit, Seitenadresse: 20 Bit, Offsetadresse: 12 Bit
Physikalische Adresse: 32 Bit, Basisadresse: 20 Bit, Offsetadresse: 12 Bit

c) Middleware für verteillte Systeme

- Eine Middleware ist eine **Softwareschicht über der Betriebssystemebene**, welche die **Entwicklung verteilter Systeme unterstützt**
- Sie soll von den **einzelnen heterogenen Systemen abstrahieren** und der **Anwendung** eine **einheitliche Schicht zur Verfügung** stellen



d) DCPS: Data-centric Publish Subscribe
Bestandteile

- Topic Anwendungsspezifischer Datentyp
- Subscriber enthält DataReader, empfängt Daten aus dem Global Data Space
- Publisher enthält DataWriter, sendet Daten in den Global Data Space
- DomainParticpants besitzen *Topics*, *Publishers,* and *Subscriber*

Konfigurierbar mit Quality of Service (QoS) Policies

Teilaufgabe 2 Thread Synchronisation

a) Mutual Exlusion Locks
   - Ein Task belegt Mutex und blockiert damit andere Tasks die zugreifen möchten, diese werden dadurch verzögert
   - Wenn Task beendet ist, dann gibt er Mutex wieder frei und andere Tasks können auf Mutex zugreifen

   Read/Write Locks
   - Gleichzeitiger Zugriff für Read-only Operationen
   - Exklusiver Zugriff für Write Operationen -> benötigt Mutex

   Semaphore
   - Mechanismus zur Synchronisierung
   - Besteht aus Zähler und zwei nicht unterbrechbare Aktionen P (Passieren) und V(Verlassen)
     - Wenn ein Task passieren möchte wird der Zähler i dekrementiert, ist i < 0 wird der Task blockiert, ist i>=0 darf der Task passieren
     - Wenn ein Task verlassen möchte wird der Zähler i inkrementiert, ist i < 1 wird der blockierte Task bereit gemacht, ist i>1 darf der Task verlassen