Contents

1 Formeln

Realzeitbedingungen:

- 1. Realzeitbedingung:
  $\rho_{max,ges} = \sum_{j=1}^{n} \frac{t_{Emax,i}}{t_{Pin,i}} \leq c$ mit c = Anzahl Rechnerkerne
- 2. Realzeitbedingung: Für alle Rechenzeitanforderungen i muss gelten:
  $t_{Dmin,j} \leq t_{Rmin,j} \leq t_{Rmax,j} \leq t_{Dmax,j}$

$t_{Pmin,i} = minimal => t_{max,i} = \frac{1}{t_{Pmin,i}}$
$t_{Pmax,i} = maximal$ <= uninteressant
$t_{Dmin,i}$ = minimal zulässige Reaktionszeit
$t_{Dmax,i}$ = maximal zulässige Reaktionszeit

- Ausführuntgszeit (Executiontime) = Rechenzeit für eine RZ-Anforderung (ohne Warte oder Schlafzeiten)
  - WCET $t_{Emax,i}$ –> Erfahrung oder Messen Worstcase
  - BCET $t_{Emin,i} = 0$ Bestcase

$T_{Rmax,i}$ = maximale Reaktionszeit
$T_{Rmin,i}$ = minimale Reaktionszeit
$T_{R,i} = t_{W,i} + t_{E,i}$ wobei $t_{W,i}$ Summe aller Wartezeiten

- Latenzzeit t_{L_i} - Interrup Latenzzeit - Tasklatenzzeit

$\rho_i = \frac{t_{E,i}}{t_{P,i}}$ Auslastung dur RZ-Anforderung i

$\rho_{max,i} = \frac{t_{Emax,i}}{t_{Pin,i}}$ Worstcase, max. Auslastung

1. RT Bedingung

$$\rho_{max,ges} = \sum_{j=1}^{n} \frac{t_{Emax,i}}{t_{Pin,i}} \leq c$$

j = für alle RZ-Anforderungen, c = Anzahl der Rechnerkerne

Technischer prozess:

- $t_{P,i}$ = Prozesszeit, zeitlicher Abstand zwischen zwei RT-Anforderungen i $\boxed{t_{Pmin,i}}$
- $t_{Dmin,i}$ = minimal zulässige Reaktionszeit
- $t_{Dmax,i}$ = maximal zulässige Reaktionszeit
- $t_{Ph,i}$ = Phase, zeitlicher Abstand zwischen zwei unterschiedlicher Ereignise

Rechenprozesse:

- $t_{Emin,i}$ = minimale Ausführungszeit BCET
- $t_{Emax,i}$ = maximale Ausführungszeit WCET
- $t_{Rmin,i}$ = minimale Reaktionszeit
- $t_{Rmax,i}$ = maximale Reaktionszeit
  ↳ Zeitlicher Abstand zwischen dem Eintreffen einer RT-Anforderung i und dem Ende der Bearbeitung
- $t_{W,i}$ = Wartezeit, Summe der Zeiten, in der eine Codesequenz arbeiten könnte, aber nicht dran kommt.

Systemsoftware:

- $t_{L,i}$ = Latenzzeit, zeitlicher Abstand zwischen dem Eintreffen einer RT-Anforderung i und dedm Start der Bearbeitung

- Schedulingverfahren

1. RT Bedingung

$$\rho_{max,ges} = \sum_{j=1}^{n} \frac{t_{Emax,j}}{t_{Pmin,j}} \leq c$$

j = für alle RZ-Anforderungen, c = Anzahl der Rechnerkerne

2. RT Bedingung

Für alle RZ-Anforderungen j muss gelten:

$t_{Dmin,j} \leq t_{Rmin,j} \leq t_{Rmax,j} \leq t_{Dmax,j}$

Utilization $u = \sum_{j=1}^{n} \frac{t_{Emax,j}}{min(t_{Dmax,j}, t_{Pmin,j})}$

2 Zeiten zeug

```
struct timespec {
        time_t    tv_sec;         /* seconds */
        long      tv_nsec;        /* nanoseconds */
};
```

- CLOCK_REALTIME
  Systemweite realzeit Uhr. Diese Uhr zu setzten erfordert Root Rechte.

- CLOCK_MONOTONIC
  Kann nicht gesetzt werden. Gibt die vergangene Zeit ab einem unbestimmten Zeitpunkt an.

```
#include <time.h>
main(int argc, char **argv)
{
        struct timespec start, end;
        clock_gettime(CLOCK_MONOTONIC, &start); /* mark start time */
        sleep(1);       /* do stuff */
        clock_gettime(CLOCK_MONOTONIC, &end);   /* mark the end time *
        int diff = diff (end.tv_sec - start.tv_sec) + end.tv_nsec - st
```

}

- Absolute Zeit: Zeit die überall gleich schnell ist
- Realtive Zeit: Zeit ist abhängig von der Geschwindigkeit und der Gravitation.

```
#include <sys/time.h>

void timeradd(struct timeval *a, struct timeval *b, struct timeval *re

void timersub(struct timeval *a, struct timeval *b,     struct timeval

void timerclear(struct timeval *tvp);

int timerisset(struct timeval *tvp);

int timercmp(struct timeval *a, struct timeval *b, CMP);
```

3 Petrie Netze

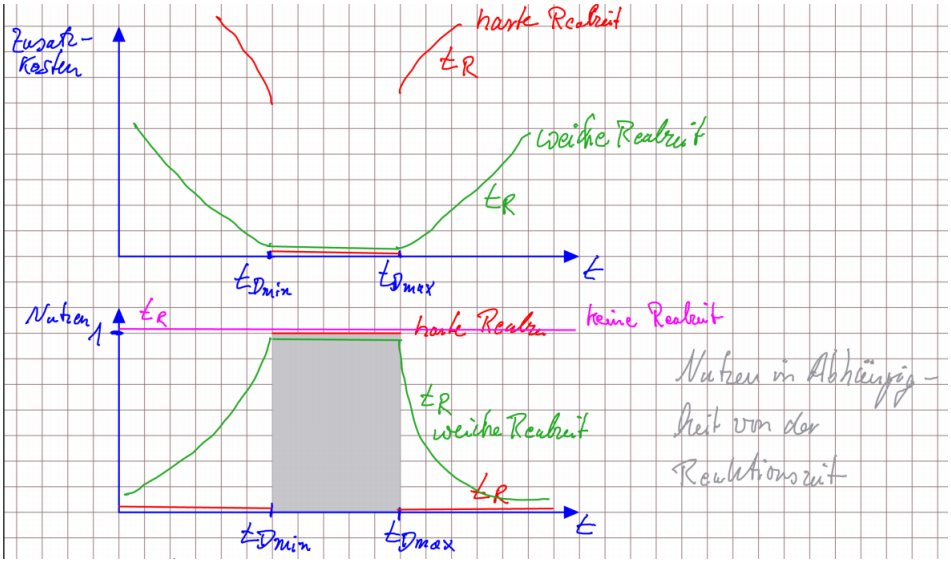
4 Datenfluss Diagram

5 Struktugram

6 Zeug

6.1 Harte und Weiche Realzeit

Harte und weiche Realzeit/Echtzeit



6.2 Bits

Oberen 4 Bits von 10101101 in Dezimal.
1010 allein stehen durch 4 mal bitshift rechts => 1010 = 0xA = 10
Bitsmaskieren:

- AND 1010 & 0111 = 0010
- OR 0101 | 0011 = 0111
- XOR 0110 $\wedge$ 1011 = 1101