64-041 Übung Rechnerstrukturen und Betriebssysteme

Aufgabenblatt 10 Ausgabe: 21.12., Abgabe: 11.01. 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 10.1 (Punkte 5+5+5)

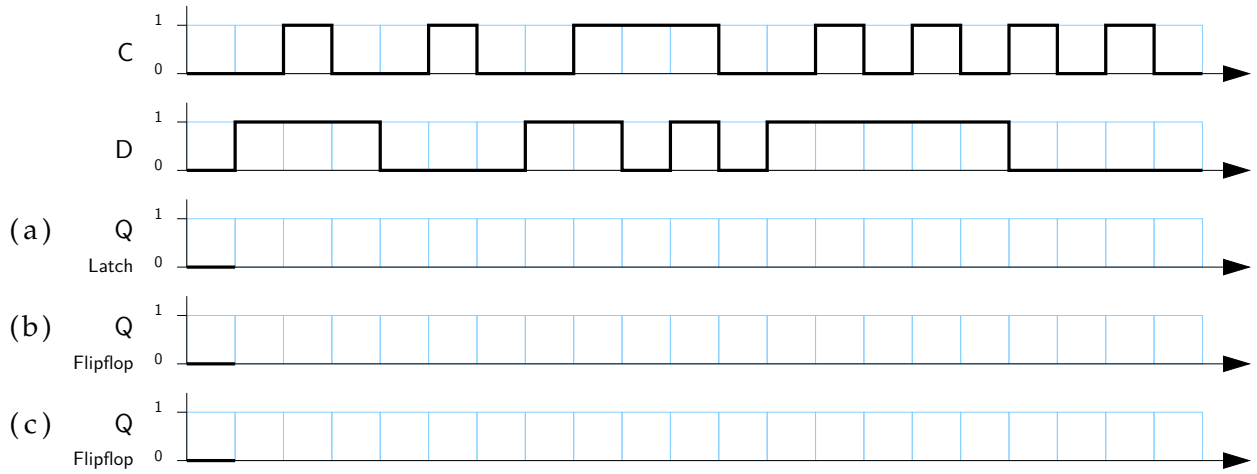
Flipflops: Vervollständigen Sie das nachfolgende Impulsdiagramm für den angegebenen Verlauf des Taktsignals C und des Eingangssignals D. Wir nehmen an, dass die Flipflops jeweils eine Zeiteinheit benötigen, bis ihr neuer Ausgangswert Q am Ausgang anliegt. Wann werden dabei Zeitbedingungen verletzt?

(a) Für ein high-aktives D-Latch (pegelgesteuert)

(b) Für ein vorderflankengesteuertes D-Flipflop

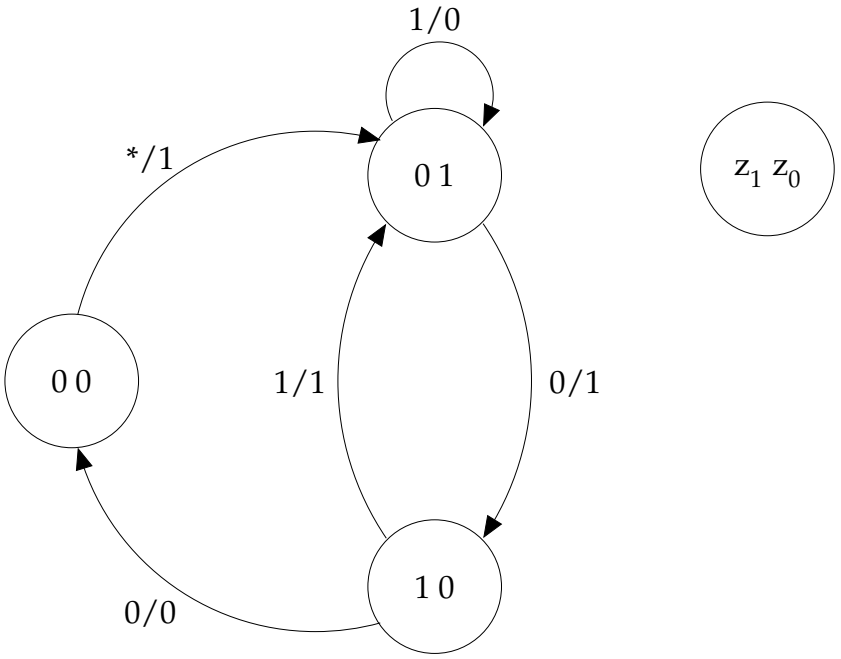
(c) Für ein zweiflankengesteuertes D-Flipflop, dessen Funktion durch folgenden Pseudocode beschrieben ist:

```
void zweiflankengesteuertes_D-FF (input C, D; output Q)
  local P;
{ if (rising_edge(C))   P= D;
  if (falling_edge(C))  Q= P;
}
```



Aufgabe 10.2 (Punkte 10+10+5+5+5)

Schaltwerk-Analyse: Wir betrachten das Zustandsdiagramm eines Automaten mit einem Eingang x, einer Ausgabe y und den drei Zuständen $Z = (z_1, z_0) = \{(0,0), (0,1), (1,0)\}$:



(a) Ermitteln Sie aus dem Zustandsdiagramm die zugehörigen Gleichungen für die Übergangsfunktion δ sowie die Ausgangsfunktion λ in disjunktiver Form.

Zur Lösung sollen dabei Übergangs- und Ausgangstabellen erstellt werden, die dann in KV-Diagramme übertragen werden und aus denen dann minimierte Schaltfunktionen ermittelt werden können.

(b) Überprüfen Sie den Automaten auf Vollständigkeit (in jedem Zustand ist für jede Eingangsbelegung mindestens ein Übergang aktiv) und Widerspruchsfreiheit (in jedem Zustand ist für jede Eingangsbelegung höchstens ein Übergang aktiv).

(c) Handelt es sich um einen Mealy- oder einen Moore-Automaten? Begründen Sie Ihre Antwort kurz.

(d) Erstellen Sie einen Schaltplan mit HADES.

(e) Ist der Automat in der vorgegeben Form praktisch einsetzbar?

Wenn ja: Warum?
Wenn nicht: Wie kann man das Problem lösen?

Aufgabe 10.3 (Punkte 5+10+10+15)

Entwurf eines Automaten: Wir betrachten einen Zähler mit einem Eingang x und der Zählfolge $\{0,2,4,6,7,5,3,1,0\ldots\}$ für den Eingangswert $x = 1$ sowie der Zählfolge $\{0,1,2,3,4,0\ldots\}$ für $x = 0$. Der Zähler startet im Zustand 0, und wechselt bei $x = 0$ von den Werten $\{5,6,7\}$ jeweils in den Zustand 0.

Zusätzlich hat der Zähler einen Ausgang *even*, der für geradzahlige Werte den Ausgangswert 1 liefert.

(a) Handelt es sich um einen Mealy- oder um einen Moore-Automaten? Begründen Sie ihre Antwort kurz.

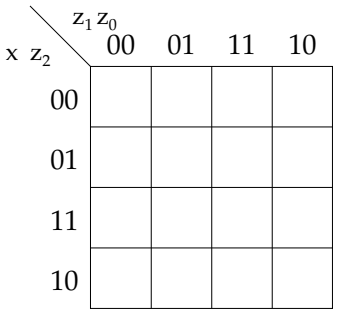
(b) Zeichnen Sie das Zustandsdiagramm des Automaten.

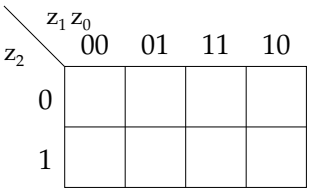
(c) Vervollständigen Sie die Zustandstabelle des Automaten. Die einzelnen Zustände Z sollen dabei im Binärcode als 3-bit Werte (z_2, z_1, z_0) codiert werden.

Ergänzen Sie die fehlenden Zustände und die zugehörigen Ausgangswerte. Die Tabelle enthält links den Eingangswert x und den aktuellen Zustand Z in Binärcodierung (z_2, z_1, z_0). Angegeben sind dann der Folgezustand Z^+ und der Ausgangswert für den Ausgang *even*.

x	z_2	z_1	z_0	z_2^+	z_1^+	z_0^+	*even*
0	0	0	0	0	0	1	1
		...			...		
1	0	0	0	0	1	0	1
		...			...		

(d) Übertragen Sie die Ausgangsfunktionen aus der obigen Zustandstabelle in KV-Diagramme und minimieren Sie die einzelnen Funktionen. Markieren Sie mögliche Schleifen und geben Sie die zugehörigen boole'schen Ausdrücke für den Folgezustand (z_2^+, z_1^+, z_0^+) und den Ausgabewert *even* in disjunktiver Form an. Dabei soll folgende Variablenbelegung genutzt werden:





Aufgabe 10.4 (Punkte 10 +5)

Installation und Test der GNU Toolchain: In Vorbereitung auf Kapitel 13 zum x86-Assembler und analog zu den Beispielen in Kapitel 2 ab Folie 92, sollen Sie selbst Zugang zu einem C-Compiler und den zugehörigen Tools haben. Wir empfehlen die *GNU Toolchain* mit dem `gcc` C-Compiler und Werkzeugen. Die Programme sind auf Linux-Systemen in der Regel vorinstalliert, so dass Sie die Befehle direkt ausführen können.

Im Informatikum können Sie die Dual-Boot Rechner in den Poolräumen unter Linux nutzen. Wenn Sie zu Hause lernen und keinen Linux PC haben, bzw. die Cygwin-Umgebung (s.u.) nicht installieren wollen, können sie auch „Remote“ auf den Informatik Rechnern arbeiten: aus einem Linux-Terminal / der Windows-Eingabeaufforderung einloggen:

```
ssh rzssh1.informatik.uni-hamburg.de
```

. . . dort dann Start der Programme. Achtung: ihr Linux HOME-Verzeichnis ist `infhome`.

Für Windows-Systeme könnten Sie die Cygwin-Umgebung von `cygwin.com` herunterladen und installieren. Im Setup von Cygwin dann bitte den `gcc`-Compiler und die Entwickler-Tools auswählen und installieren. Natürlich können Sie auch jeden anderen C-Compiler verwenden, müssen sich dann aber die benötigten Befehle und Optionen selbst heraussuchen.

Hauptsächlich soll Sie die Aufgabe dazu motivieren vielleicht auf dem eigenen Rechner mit den Werkzeugen zu „spielen“ und so auch selbst zu sehen was aus programmiertem Code auf niedrigeren Abstraktionsebenen wird. Also installieren Sie die entsprechenden Programme (distributionsabhängig).

Anmerkung: Keine Angst, die Aufgabe soll zeigen, wie Assemblercode aussieht und Ihnen helfen erste Einblicke zu gewinnen, wie Betriebssystem, (Programm-) Binär-Code und die Hardware zusammenspielen. **Es geht nicht darum Assemblerprogrammierung zu lernen!**

Für einen ersten Test tippen Sie bitte das folgenden Programm ab oder laden Sie sich die Datei `aufg10_4.c` herunter. Passen Sie die Datei an, indem Sie dort ihren Namen und die Matrikelnummer eintragen. Anschließend sollen Sie das Programm übersetzen und sich den erzeugten Assembler- und Objektcode analysieren.

```
1  /* aufg10_4.c
2   * Einfaches Programm zum Test des C-Compilers und der zugehörigen Tools.
3   * Bitte setzen Sie in das Programm ihren Namen und die Matrikelnummer ein
4   * und probieren Sie alle der folgenden Operationen aus:
5   *
6   * Funktion          Befehl                                    erzeugt
7   * ----------------+-----------------------------------------+----------------
8   * C -> Assembler:  gcc -O2 -S aufg10_4.c                    -> aufg10_4.s
9   * C -> Objektcode: gcc -O2 -c aufg10_4.c                    -> aufg10_4.o
10  * C -> Programm:   gcc -O2 -o aufg10_4.exe aufg10_4.c       -> aufg10_4.exe
11  * Disassembler:    objdump -d aufg10_4.o
12  *                  objdump -d aufg10_4.exe
13  * Ausführen:       aufg10_4.exe
14  */
15
```

```
16 #include <stdio.h>
17
18 int main( int argc, char** argv )
19 { int matrikelNr    = 123456;
20   char vorname[32]  = "John";
21   char nachname[32] = "Doe";
22 //char *vorname     = "John";
23 //char *nachname    = "Doe";
24
25   printf("Name: %s %s - Matrikelnr.: %d\n", vorname, nachname, matrikelNr);
26   return 0;
27 }
```

(a) Machen Sie sich mit dem Compiler und den Tools vertraut. Probieren Sie die vorgeschlagenen Befehle aus und sehen Sie sich die Ausgaben an.

Erzeugen Sie eine Textdate, die die Ausgabe des Programms und ein Listing des Disassemblers enthält. Dies geschieht am einfachsten mit den folgenden Befehlen:

```
./aufg10_4.exe          > loesung10_4.txt
echo "=========================================" >> loesung10_4.txt
objdump -d aufg10_4.o >> loesung10_4.txt
```

Markieren Sie in der Datei an welcher Stelle des Codes: Vorname, Nachname und Matrikelnummer stehen (mit kurzer Begründung). Diese Datei ist als Lösung des Aufgabenteils abzugeben.

(b) optionale Zusatzpunkte

In dem Code `aufg10_4.c` sind die Zeilen 22 und 23 auskommentiert. Ändern Sie die Variablen für Vor- und Nachnamen in die zweite Version (Zeiger auf den String, statt `char`-Array).

Was ändert sich in dem Assembler-Code? Es genügt, die Änderungen (inhaltlich) zu beschreiben, es müssen keine Listings abgegeben werden.