64-041 Übung Rechnerstrukturen

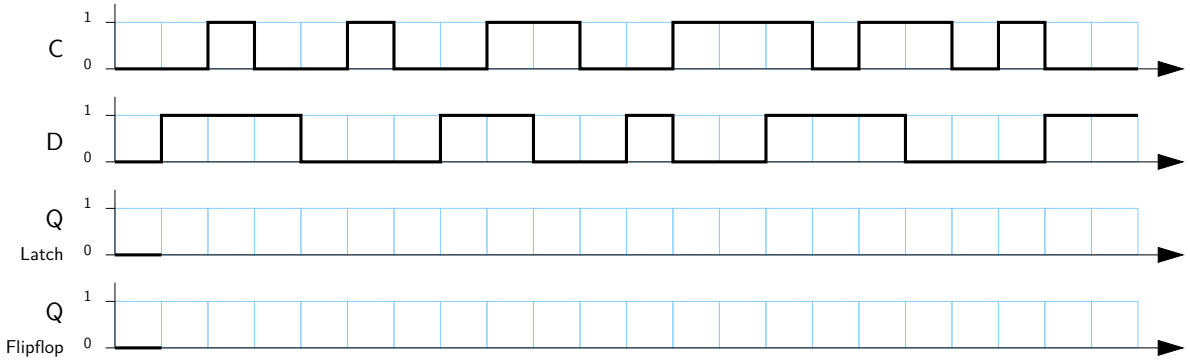
Aufgabenblatt 10 Ausgabe: 20.12., Abgabe: 10.01. 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 10.1 (Punkte 10+10)

D-Latch und D-Flipflop: Wir betrachten das pegelgesteuerte D-Latch (*high-aktiv*) und das vorderflankengesteuerte D-Flipflop. Wir nehmen an, dass die beiden Flipflops jeweils eine Zeiteinheit benötigen, bis ihr neuer Ausgangswert Q am Ausgang anliegt.

Vervollständigen Sie das Impulsdiagramm für den angegebenen Verlauf des Taktsignals C und des Eingangssignals D. Wann werden dabei Zeitbedingungen verletzt?

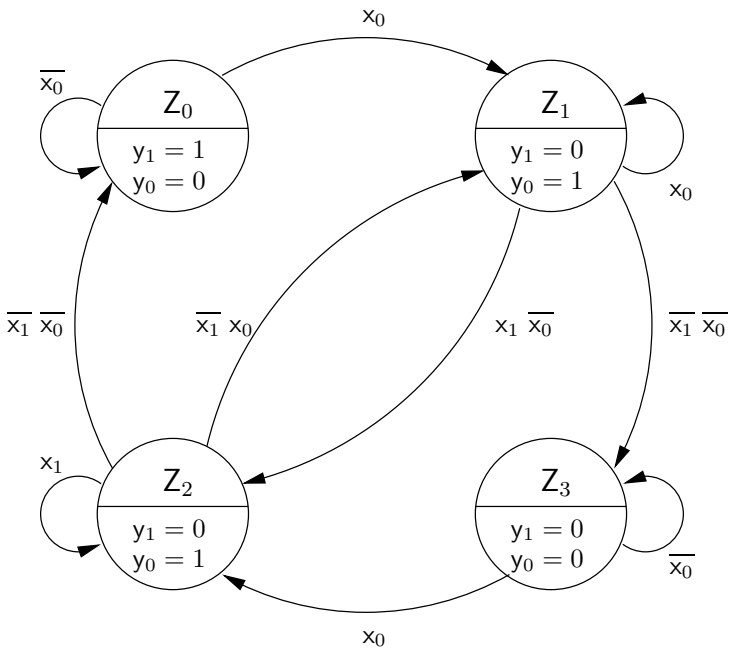


Aufgabe 10.2 (Punkte 10+10+10)

Schaltwerk-Analyse: Wir betrachten das Zustandsdiagramm eines Moore-Schaltwerks mit Eingängen $X = (x_0, x_1)$ und Ausgaben $Y = (y_0, y_1)$ sowie vier Zuständen Z_0, Z_1, Z_2, Z_3. Wir codieren die Zustände Z binär mit zwei Bits (z_1, z_0) und damit $Z_0 = (0,0)$, $Z_1 = (0,1)$, $Z_2 = (1,0)$ und $Z_3 = (1,1)$.

(a) Ermitteln Sie aus dem Zustandsdiagramm die zugehörigen Gleichungen für die Übergangsfunktion δ zur Berechnung des Folgezustands Z^+ aus aktuellem Zustand Z und den Eingabewerten X.

Eine Lösungsmöglichkeit ist das Aufstellen der Flusstafel, alternativ das Aufstellen der Übergangs- und Ausgangstabellen und dann die Logikminimierung.



(b) Ermitteln Sie die zugehörigen Gleichungen für die Ausgangsfunktion λ zur Berechnung des Ausgangswerts Y als Funktion des aktuellen Zustands Z.

(c) 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).

Aufgabe 10.3 (Punkte 10+10+10)

Entwurf eines Schaltwerks: Wir betrachten ein Schaltwerk mit sechs Zuständen $Z_0, \ldots Z_5$, einem Eingang x und vier Ausgängen y_1, y_2, y_3, y_4. Die Zustandsübergänge und die Ausgabe sind dabei durch folgende Tabelle gegeben:

x	Z	Z^+	y_1	y_2	y_3	y_4
0	Z_0	Z_1	0	1	0	0
0	Z_1	Z_2	0	0	0	0
0	Z_2	Z_3	1	0	0	0
0	Z_3	Z_5	0	1	1	1
0	Z_4	Z_0	0	0	1	0
0	Z_5	Z_4	1	1	0	1
1	Z_0	Z_2	0	1	0	0
1	Z_1	Z_4	0	0	0	0
1	Z_2	Z_0	1	0	0	0
1	Z_3	Z_4	0	1	1	1
1	Z_4	Z_3	0	0	1	0
1	Z_5	Z_2	1	1	0	1

(a) Zeichnen Sie das Zustandsdiagramm des Schaltwerks.

(b) Um das Schaltwerk zu realisieren, wählt man jetzt eine Codierung der Zustände. Wir betrachten zwei von vielen Möglichkeiten:
Bestimmen Sie für beide Codierungen die Funktionen des Zustandsübergangsschaltnetzes (das δ-Schaltnetz). Beachten Sie dabei die Möglichkeit von *Don't-Cares*. Die Tabellen und KV-Diagramme sollen mit abgegeben werden.

Z_i	Codierung 1 ($z_2\ z_1\ z_0$)	Codierung 2 ($z_2\ z_1\ z_0$)
Z_0	(100)	(001)
Z_1	(000)	(110)
Z_2	(011)	(000)
Z_3	(001)	(101)
Z_4	(010)	(100)
Z_5	(101)	(010)

(c) Offenbar lässt sich durch eine geeignete Codierung der Zustände eine erhebliche Vereinfachung der Schaltfunktionen erreichen. Das Problem ist nur, dass es alles andere als einfach ist, eine bestmögliche Codierung zu finden, wobei man dann auch noch die Funktionen für die Ausgabe (das λ-Schaltnetz) mit berücksichtigen müsste.

Geben Sie eine Codierung für die sechs Zustände an, die zumindest das λ-Schaltnetz des Automaten minimiert. Es sind dabei auch mehr als drei Bits für die Zustandscodes erlaubt. Erläutern Sie ihre Vorgehensweise.

Aufgabe 10.4 (Punkte 10+10)

Installation und Test der GNU Toolchain: Ziel dieser Aufgabe ist es, dass Sie selbst Zugang zu einem C-Compiler und den zugehörigen Tools haben. Wir empfehlen die *GNU Toolchain* mit dem `gcc` C-Compiler und Werkzeugen. Diese ist auf den meisten Linux-Systemen bereits vorinstalliert, so dass Sie die Befehle direkt ausführen können.

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

Als komfortable Alternative ein einfaches lauffähiges System zu erhalten, können Sie eine virtuelle Maschine von der RS-Webseite herunterladen und auf Ihrem Rechner in Betrieb nehmen. Hades und die `gcc` Toolchain sind dort passend vorkonfiguriert. Zur Ausführung der virtuellen Maschine (im OVF-Format), können Sie als kostenlose Software sowohl Oracle VirtualBox (Windows, Linux, MacOS), als auch VMware Player (Windows, Linux) nutzen.

Prinzipiell kann auch auf die Rechner in den PC-Poolräumen zurückgegriffen werden, sie sind als Dual-Boot Systeme auch mit Ubuntu 16.04 ausgestattet.[1]

Für einen ersten Test tippen Sie bitte das folgenden Programm ab oder laden Sie sich die Datei `aufg10_4.c` von der Webseite herunter. Passen Sie die Datei an, indem Sie dort ihre Matrikelnummer eintragen. Anschließend können Sie das Programm übersetzen und sich den erzeugten Assembler- und Objektcode anschauen.

[1] Der Parameter `-m32` funktioniert hier aber nicht, da keine 32-bit Bibliotheken installiert wurden. Auf den Rechnern bei TAMS in F-304 sollte alles funktionieren; wegen Aufschließen nachfragen...

```
/* aufg10_4.c
 * Einfaches Programm zum Test des gcc-Compilers und der zugehörigen Tools.
 * Bitte setzen Sie in das Programm ihre Matrikelnummer ein und probieren
 * Sie alle der folgenden Operationen aus:
 *
 * Funktion         Befehl                                  erzeugt
 * ----------------+---------------------------------------+----------------
 * C -> Assembler:  gcc -O2 -S aufg10_4.c                  -> aufg10_4.s
 * C -> Objektcode: gcc -O2 -c aufg10_4.c                  -> aufg10_4.o
 * C -> Programm:   gcc -O2 -o aufg10_4.exe aufg10_4.c     -> aufg10_4.exe
 * Disassembler:    objdump -d aufg10_4.o
 * Ausführen:       aufg10_4.exe
 *
 * 32bit Code auf 64bit System: gcc -m32 ...
 */

#include <stdio.h>

int main( int argc, char** argv )
{ int matrikelNr = 123456;

  printf( "Meine Matrikelnummer ist %d (0x%x)\n", matrikelNr, matrikelNr );
  return 0;
}
```

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

Hinweis: auf x86-64 Systemen (64bit Linux) können Sie auch die `gcc`-Compileroption `-m32` ausprobieren, um 32bit Code zu erzeugen.

(b) Schicken Sie den Quellcode sowie den erzeugten Assemblercode und die Ausgabe des Befehls `objdump -d` (GNU Toolchain) an Ihren Gruppenleiter.

Bei Verwendung anderer Compiler und Tools bitte ebenfalls die entsprechenden Ausgabedateien generieren und einschicken.

Hinweis: den erzeugten Programmcode (`aufg10_4.exe`) nicht mit abgeben, da verschiedene Mailserver Mails mit angehängten ausführbaren Programmen wegen eventuell enthaltener Viren automatisch zurückhalten.