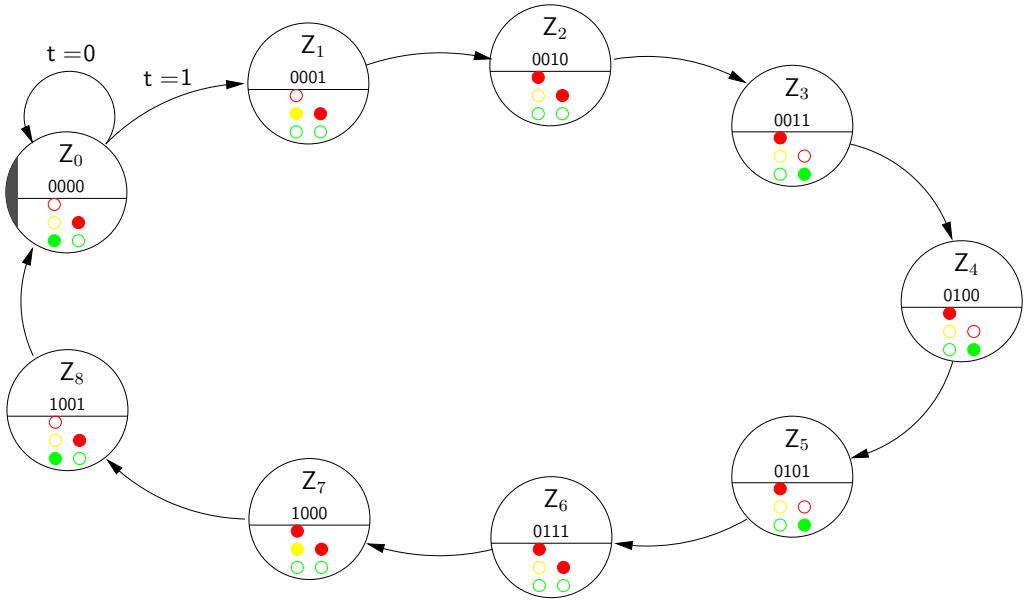
64-041 Übung Rechnerstrukturen

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

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 10.1 (Punkte 10+10+10+15)

Gekoppelte Automaten: In Aufgabe 9.3 (letzte Woche) sollte eine Ampelschaltung („Fußgänger Bedarfsampel“) als Moore-Automat entworfen werden. Um die jeweiligen Grünphasen länger zu machen, sollten in dem Automaten aufeinander folgende Zustände ($Z_3 \ldots Z_5$, bzw. Z_8 und Z_0) genutzt werden. Die folgende Grafik zeigt das Zustandsdiagramm der Musterlösung:



Wird der Automat jetzt aber mit einem Takt von 1 KHz betrieben und sollen Ampelschaltungen mit einigen Sekunden Dauer bei Rot und Gelb sowie 30 Sekunden bei Grün realisiert werden, dann ist es viel zu umständlich 30 000 Takte für die Grünphase zu codieren. Üblicherweise setzt man deshalb zwei (oder mehr) *gekoppelte Automaten* ein: einen „Hauptautomaten“ der die Zustandsübergänge realisiert und einen Zähler (trivialer Automat), der für die Wartezeiten sorgt.

(a) Beschreiben Sie (textuell), wie diese Automaten zusammenarbeiten. Welche Leitungen gehen vom Hauptautomaten zum Zähler, welche gehen zurück?

(b) Überlegen Sie sich was passiert, wenn beide Automaten mit identischem Taktsignal arbeiten: der Taktvorderflanke? Was ist dabei zu berücksichtigen? Beschreiben Sie, wie die Fußgängerampel auf die Grünphase schaltet (Z_3), die Wartezeit von 30 Sekunden vergeht und anschließend dieser Zustand verlassen wird.

Tipp: bei den gekoppelten Automaten und ihren Taktschemata ist entscheidend, wann die einzelnen Zustandsübergänge stattfinden. Bei gegenseitigen Abhängigkeiten (Steuersignale, Zählerstände) kann es leicht vorkommen, dass man einen Takt „zu spät“ ist.

(c) Überlegen Sie sich was passiert, wenn die Automaten unterschiedlich getaktet sind, so dass der eine mit der Taktvorderflanke und der zweite Automat mit der Rückflanke arbeitet. Verfahren Sie wie in Aufgabenteil (b): beschreiben Sie, wie die Ampel auf Grünphase schaltet, die Wartezeit vergeht und anschließend der Zustand Z_3 verlassen wird.

(d) Entwerfen Sie eine einfache Ampelschaltung, ähnlich dem Beispiel aus der Vorlesung: *Abschnitt 12.9.1: „Schaltwerke – Beispiele – Ampelsteuerung“*.

Es gibt einen zentralen Takt von 1 KHz für beide Automaten. Die Ampel soll zyklisch die vier Ausgaben {rot, rot-gelb, grün, gelb} erzeugen, wobei folgende Zeitbedingungen einzuhalten sind:

Zustand	Zeitdauer
rot	25 Sek.
rot-gelb	2 Sek.
grün	30 Sek.
gelb	4 Sek.

Zeichnen Sie das Zustandsdiagramm für den Hauptautomaten als Moore-Modell und geben sie für jeden Zustand an, welche Werte die Ausgangsleitungen (die Lampen und alle Steuerleitungen für den/die Zähler) haben. Als Taktschema können Sie eine der beiden Varianten Aufgabenteil (b) oder (c) wählen.

Aufgabe 10.2 (Punkte 10+10+15)

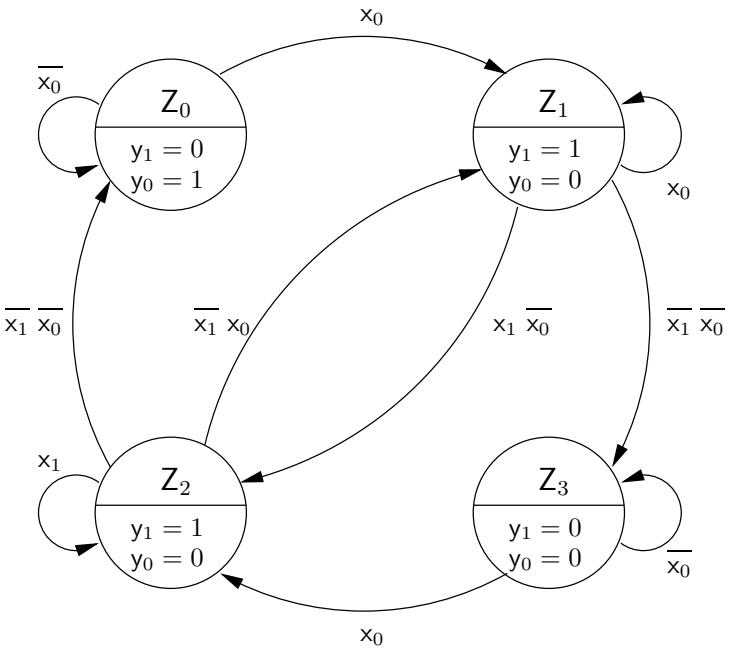
Schaltwerk-Analyse: Wir betrachten das Zustandsdiagramm eines Moore-Schaltwerks mit Eingängen $X = (x_1, x_0)$ und Ausgaben $Y = (y_1, y_0)$ sowie vier Zuständen Z_0, Z_1, Z_2, Z_3. Diese sind binär mit zwei Bits (z_1, z_0) codiert und damit $Z_0 = (0,0)$, $Z_1 = (0,1)$, $Z_2 = (1,0)$ und $Z_3 = (1,1)$.

(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)

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 unter Windows oder Linux in Betrieb nehmen. Hades und die `gcc`-Toolchain sind dort passend vorkonfiguriert.

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]

[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...

Für einen ersten Test tippen Sie bitte das folgenden Programm ab oder laden Sie sich die Datei `aufg10_3.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.

```
/* aufg10_3.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_3.c                  -> aufg10_3.s
 * C -> Objektcode: gcc -O2 -c aufg10_3.c                  -> aufg10_3.o
 * C -> Programm:   gcc -O2 -o aufg10_3.exe aufg10_3.c     -> aufg10_3.exe
 * Disassembler:    objdump -d aufg10_3.o
 * Ausführen:       aufg10_3.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_3.exe`) nicht mit abgeben, da verschiedene Mailserver Mails mit angehängten ausführbaren Programmen wegen eventuell enthaltener Viren automatisch zurückhalten.