64-041 Übung Rechnerstrukturen

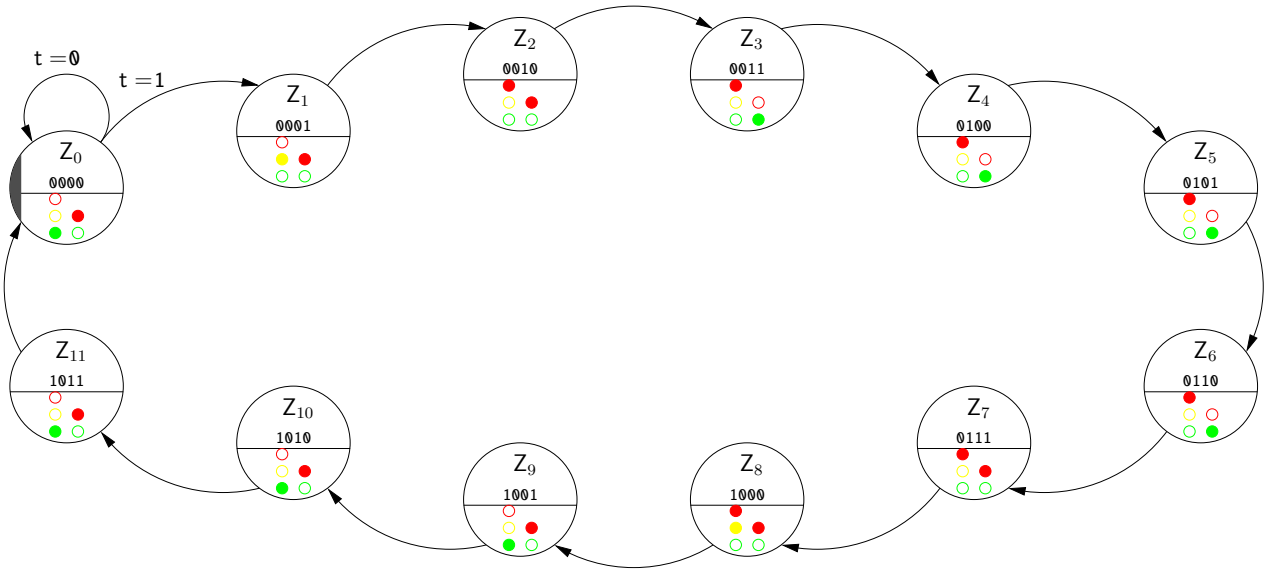
Aufgabenblatt 9 Ausgabe: 23.12., Abgabe: 13.01. 12:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 9.1 (Punkte 10+10+10)

Gekoppelte Automaten: In Aufgabe 8.4 (letzte Woche) sollte eine Ampelschaltung („Fußgänger Bedarfsampel“) als Moore-Automat entworfen werden. Um die jeweiligen Grünphasen länger zu machen, sollten in dem Automaten vier aufeinander folgende Zustände ($Z_3 \ldots Z_6$, bzw. $Z_9 \ldots Z_{11}$+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) Beide Automaten sollen mit identischem Taktsignal arbeiten. Was ist dabei zu berücksichtigen? Tipp: Abhängigkeiten untereinander, siehe Teil (a)

(c) Entwerfen Sie nach diesem Schema eine einfache Ampelschaltung, ähnlich dem Beispiel aus der Vorlesung *Kapitel 14: „Schaltwerke – Beispiele“*.

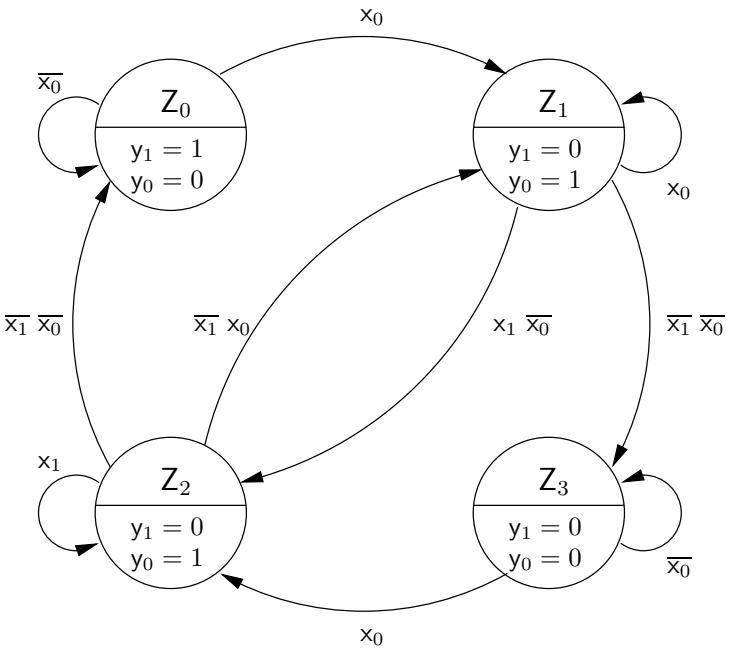
Es gibt einen zentralen Takt von 1 KHz für beide Automaten. Die Ampel soll zyklisch die vier Ausgaben {rot, rot-gelb, grün, gelb} erzeugen, wobei folgende Zeitbedingungen einzuhalten sind:

Zustand	Zeitdauer
rot	20 Sek.
rot-gelb	3 Sek.
grün	30 Sek.
gelb	5 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.

Aufgabe 9.2 (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 9.3 (Punkte 10+10)

CMOS-Komplexgatter: Überlegen Sie sich die Struktur eines Komplexgatters zur Berechnung der folgenden Funktion:

$$y = \overline{(a \wedge b) \vee ((c \vee d) \wedge (e \vee f))}$$

(a) Beschreiben Sie den Aufbau und zeichnen Sie den Schaltplan für das Gatter (mit den vereinfachten Transistorsymbolen für n-Kanal und p-Kanal Transistoren).

(b) Wie viele Transistoren benötigt das Komplexgatter? Wie viele Transistoren würde demgegenüber die herkömmliche Realisierung aus mehreren CMOS-Gattern benötigen?

Aufgabe 9.4 (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 mit installieren. Alternativ können Sie auch einen anderen C-Compiler verwenden, Sie müssen sich dann aber die benötigten Befehle und Optionen selbst heraussuchen.

Für einen ersten Test tippen Sie bitte das folgenden Programm ab, oder laden Sie sich die Datei `aufgabe9.c` von der Webseite zur Vorlesung herunter. Die Syntax von C ist der Syntax von Java sehr ähnlich.

Ändern Sie dann die Datei, indem Sie ihre Matrikelnummer eintragen. Übersetzen Sie das Programm und schauen Sie sich den erzeugten Assembler- und Objektcode an.

```
/* aufgabe9.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 aufgabe9.c                  -> aufgabe9.s
 * C -> Objektcode: gcc -O2 -c aufgabe9.c                  -> aufgabe9.o
 * C -> Programm:   gcc -O2 -o aufgabe9.exe aufgabe9.c     -> aufgabe9.exe
 * Disassembler:    objdump -d aufgabe9.o
 * Ausführen:       aufgabe9.exe
 */

#include <stdio.h>

int main( int argc, char** argv )
{ int matrikelnummer = 7654321;

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

(a) Machen Sie sich mit dem Compiler und den Tools vertraut.

(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: der erzeugte Programmcode (aufgabe9.exe) muss nicht mit abgegeben werden. Verschiedene Mailserver halten Mails mit angehängten ausführbaren Programmen wegen eventuell enthaltener Viren automatisch zurück.