Aufgabenblatt 8

Ausgabe 13/12/2010, Abgabe bis 03/01/2011 12:00

Name(n):
Matrikelnummer(n):
Übungsgruppe:

Aufgabe 8.1 Zeitverhalten von Addierern (5+10+5 Punkte)

Wir untersuchen das Zeitverhalten der in der Vorlesung vorgestellten Addierer (ripple-carry, carry-lookahead, carry-select). Als Zeitmodell nehmen wir eine Verzögerung von jeweils einer Zeiteinheit für den Volladdierer, einen Multiplexer und alle beim Carry-Lookahead Addierer verwendeten Teilschaltungen (Sum, CLA) an. Unter diesen Annahmen beträgt die Verzögerung für einen n-bit Ripple-Carry Addierer n Zeitschritte, da das Carry-Signal alle n Stufen durchlaufen muss, bis das höchste Bit der Summe berechnet werden kann.

a) Welche Verzögerung ergibt sich bei n Bit für den in der Vorlesung beschriebenen Carry-Lookahead-Addierer? (Dabei werden zunächst von den Sum-Blöcken die generate- und propagate Werte berechnet, dann der CLA-Baum bis zur Wurzel durchlaufen, und dann die carry-Werte zurück zu den Sum-Blöcken übertragen.)

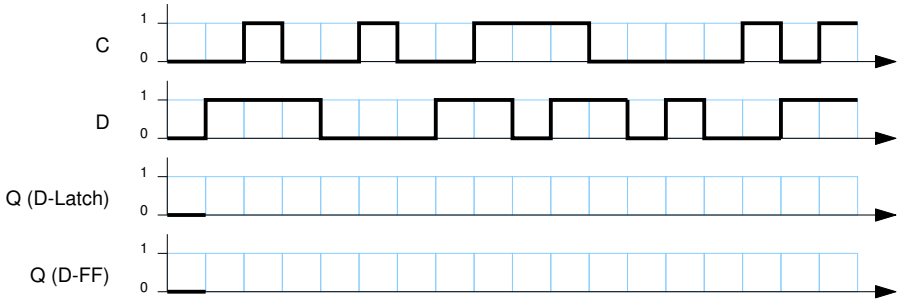
b) Für den n-bit Carry-Select Addierer wählen wir zunächst eine Aufteilung in m Blöcke von jeweils n/m bits. (Falls n/m nicht ganzzahlig ist, werden einige Blöcke um jeweils 1 Bit erweitert, bis es passt.) Wie viele Zeitschritte benötigt dieser Addierer als Funktion von n und m? Wie muss m gewählt werden, um die Verzögerung zu minimieren?

c) Geben Sie die Verzögerung für alle drei Addierer für jeweils $n = 64$ (z.B. Java `long`) und $n = 256$ (z.B. Java3D Koordinaten) an. Welche maximale Taktfrequenz ist mit den jeweiligen Addierern erreichbar, wenn wir einen Wert von 100 ps als Zeitverzögerung einer Stufe annehmen?

Aufgabe 8.2 D-Latch und D-Flipflop (10+10 Punkte)

Wir betrachten das pegelgesteuerte D-Latch und das flankengesteuerte 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.



Aufgabe 8.3 Entwurf eines Automaten (10+10+20 Punkte)

Wir betrachten einen Zähler mit einem Eingang a und der Zählfolge $\{0, 2, 4, 6, 7, 5, 3, 1, 0\}$ für den Eingangswert $a = 1$ sowie der Zählfolge $\{0, 1, 2, 3, 4, 0\}$ für $a = 0$. Der Zähler startet im Zustand 0, und wechselt bei $a = 0$ 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.

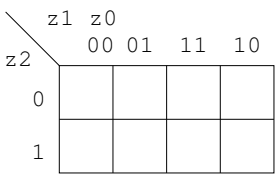
a) Zeichnen Sie das Zustandsdiagramm des Automaten.

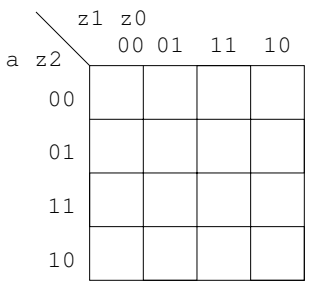
b) Vervollständigen Sie die Zustandstabelle des Automaten. Die einzelnen Zustände Z sollen dabei im Binärcode als 3-bit Werte (z_2, z_1, z_0) codiert werden.

Ergänzen Sie die fehlenden Zustände und die zugehörigen Ausgangswerte. Die Tabelle enthält links den Eingangswert a 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 $even$.

a	z_2	z_1	z_0	z_2^+	z_1^+	z_0^+	$even$
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
1	0	0	0	0	1	0	1

c) 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 Ausdrücke für den Folgezustand (z_2^+, z_1^+, z_0^+) und den Ausgabewert $even$ in disjunktiver Form an.





Aufgabe 8.4 Installation und Test der GNU-Toolchain (10+10 Punkte)

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 `www.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 `template.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.

```
/* template.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:
 *
 * C -> Assembler:  gcc -O2 -S template.c  (erzeugt template.s)
 * C -> Objektcode: gcc -O2 -c template.c  (erzeugt template.o)
 * C -> Programm:   gcc -O2 -o template.exe template.c  (erzeugt template.exe)
 * Disassembler:    objdump -d template.o
 * Ausführen:       template.exe
 */

#include <stdio.h>

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

  printf( "Meine Matrikelnummer ist %ld\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 (template.exe) muss nicht mit abgegeben werden. Verschiedene Mailserver halten Mails mit angehängten ausführbaren Programmen wegen eventuell enthalteter Viren automatisch zurück.