TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Technische Informatik

Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

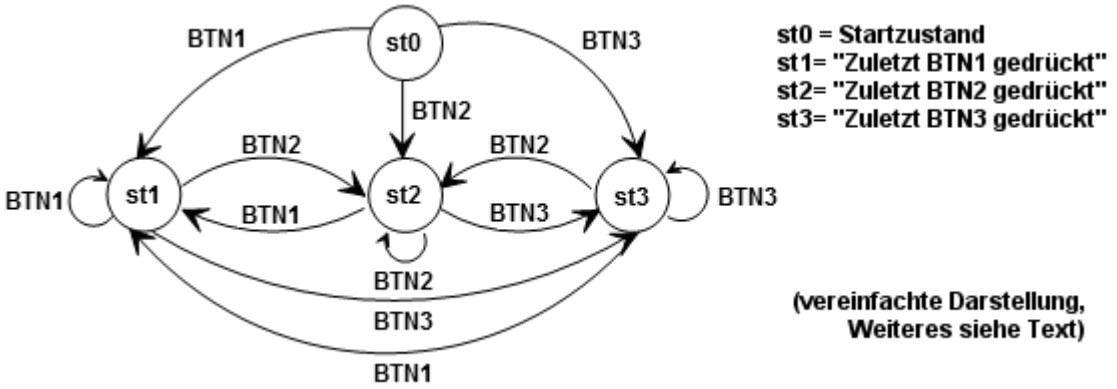
Praktikum Programmierbare Schaltkreise WS 2009/10
Komplex: Codeschloss

Ansprechpartner
Martin Zabel
martin.zabel@tu-dresden.de
INF/1095

18. Dezember 2009

Aufgabe 1

Gegeben ist der nachfolgende Zustandsgraph eines Automaten, der auf einem PLD mittels VHDL zu realisieren ist. Der Startzustand `st0` soll dabei automatisch zu Beginn der Simulation als auch mit dem „Power-Up" des PLD eingenommen werden. Eine Rückkehr in den Startzustand ist erst später vorgesehen. Wird kein Taster gedrückt, so ist im aktuellen Zustand zu verbleiben. Ebenso, wenn mehrere Taster gleichzeitig gedrückt werden.



Die Zustände sind symbolisch über eine `type`-Anweisung zu definieren, sodass die Logik-Synthese eine geeignete Kodierung wählen kann. Der aktuelle Zustand ist über die LEDs mit folgender Kodierung auszugeben:

Zustand	LD1	LD2	LD3
st0	0	0	0
st1	1	0	0
st2	0	1	0
st3	0	0	1

a) Erstellen Sie einen SM-Chart, der den Automatengraphen plus die im Text genannten Randbedingungen umsetzt.

b) Implementieren Sie den Automaten mittels VHDL und überprüfen Sie die korrekte Funktion im Simulator. Implementieren Sie dazu eine geeignete Testbench.

c) Erstellen Sie eine passende UCF-Datei für das Praktikumsboard. Für Ein- und Ausgabe sind die Taster und LEDs des Erweiterungsboards zu nutzen.

Synthetisieren Sie ihr VHDL-Modell und beheben Sie aufgetretene Fehler und Warnungen. Überprüfen Sie die korrekte Funktion des Automaten auf dem Praktikumsboard.

Werten Sie im Praktikumsprotokoll die benötigten Ressourcen aus dem *Fit Report* aus.

d) In der Standardeinstellung wählt die Logiksynthese automatisch eine „günstige" Kodierung aus. Diese kann über die Option *Synthesize Properties* ⇒ *HDL Options* ⇒ *FSM Encoding Algorithm* geändert werden.

Überprüfen Sie mittels Synthese, für welche der drei Kodierungen: One-Hot, Gray-Code und Binärcode (*Sequential*), die wenigstens Makrozellen benötigt werden.

Erläutern Sie ihre Beobachtungen im Praktikumsprotokoll.

e) Erweitern Sie den Automaten so, dass durch Betätigung des Reset-Schalters `SW1` der Startzustand eingenommen wird. Die FFs sind direkt im getakteten Prozess synchron zum Takt zurückzusetzen, sodass das Reset Vorrang vor allen anderene Eingaben hat.

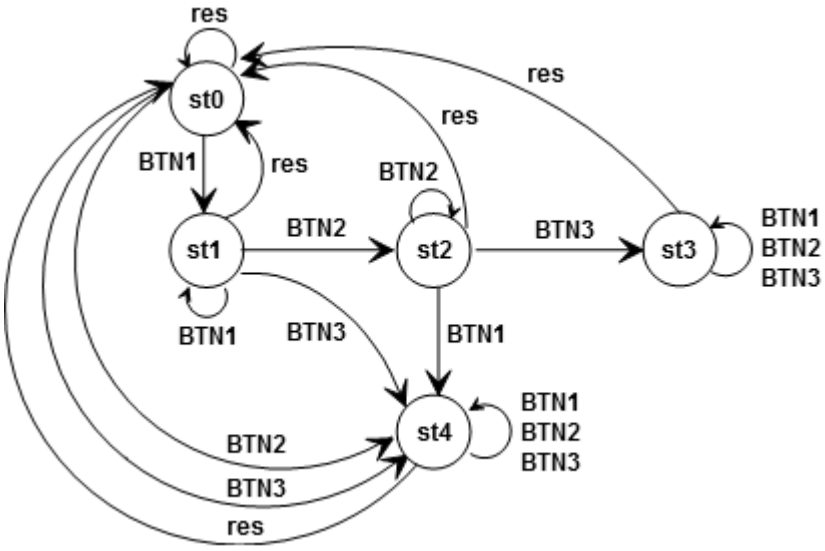
Überprüfen Sie die geänderte Funktionalität sowohl im Simulator als auch auf dem Praktikumsboard analog b) und c). Bestimmen Sie die für das Reset benötigten Ressourcen für alle Kodierungen aus d).

f) Alternativ zur Standardlösung aus e) kann das (synchrone) Reset auch so implementiert werden, dass der Reset-Schalter als „normale" Eingabe in den Zustandsübergängen behandelt wird.

- **i)** Erweitern Sie Ihren SM-Chart aus a). Der Reset-Schalter habe dabei Vorrang vor allen anderen Eingaben.
- **ii)** Implementieren Sie das Reset entsprechend im VHDL-Modell. Überprüfen Sie die geänderte Funktionalität sowohl im Simulator als auch auf dem Praktikumsboard analog b) und c). Bestimmen Sie die für das Reset benötigten Ressourcen für alle Kodierungen aus d).
- **iii)** Welche Vor- oder Nachteile sind im Entwurf sowie in der Implementierung zu erkennen.

Aufgabe 2

Gegeben ist der nachfolgende Zustandsgraph eines Automaten, der auf einem PLD mittels VHDL zu realisieren ist. Der Startzustand `st0` soll dabei automatisch mit dem „Power-Up" des PLD als auch durch Betätigung des Reset-Schalters (`RES`) eingenommen werden. Der Reset hat Vorrang vor allen anderen Eingaben und soll synchron zum Takt erfolgen. Bei keiner Eingabe oder mehreren Tastern gleichzeitig ist im aktuellen Zustand zu verbleiben.



Für den Reset-Schalter (Signalname `RES`) ist SW1 auf dem Erweiterungsboard zu verwenden.

Die Zustände sind symbolisch über eine `type`-Anweisung zu definieren, sodass die Logik-Synthese eine geeignete Kodierung wählen kann. Der aktuelle Zustand ist über die LEDs mit folgender Kodierung auszugeben:

Zustand	LD1	LD2	LD3
st0	0	0	0
st1	1	0	0
st2	0	1	0
st3	1	1	0
st4	1	1	1

a) Analysieren Sie die Funktion des Automaten:

- Erstellen Sie einen SM-Chart, der den Automatengraphen plus die im Text genannten Randbedingungen umsetzt.
- Welche Eingabe muss erfolgen, damit Zustand st1, st2 sowie st3 erreicht werden?
- Bei welchen Eingaben wird Zustand st4 erreicht?
- Welche Bedeutung besitzen die Zustände st3 und st4 im Sinne eines Akzeptors?

b) Implementieren Sie den Automaten mittels VHDL und überprüfen Sie die korrekte Funktion im Simulator. Implementieren Sie dazu eine geeignete Testbench.

c) Erstellen Sie eine passende UCF-Datei für das Praktikumsboard. Für Ein- und Ausgabe sind die Taster und LEDs des Erweiterungsboards zu nutzen.

Synthetisieren Sie ihr VHDL-Modell und beheben Sie aufgetretene Fehler und Warnungen. Überprüfen Sie die korrekte Funktion des Automaten auf dem Praktikumsboard.

Werten Sie im Praktikumsprotokoll die benötigten Ressourcen aus dem *Fit Report* aus.

d) Erweitern Sie ihr VHDL-Modell so, dass das Erreichen des Zustandes st3 über LD8 und des Zustandes st4 über LD7 angezeigt wird.

Überprüfen Sie die geänderte Funktionalität sowohl im Simulator als auch auf dem Praktikumsboard analog b) und c). Bestimmen Sie die hierfür benötigten Ressourcen.

e) Die Taster auf dem Erweiterungsboard sind bereits entprellt. Erweitern Sie den *Automaten* aus d) so, dass bei mehrmaligen Drücken derselben Taste ebenfalls der Zustand st4 eingenommen wird.

Überprüfen Sie die geänderte Funktionalität sowohl im Simulator als auch auf dem Praktikumsboard analog b) und c). Bestimmen Sie die hierfür benötigten Ressourcen.

f) Die Taster auf dem Erweiterungsboard sind bereits entprellt. Erweitern Sie den *Automaten* aus e) so, dass bei gleichzeitigem Drücken mehrerer Tasten der Zustand st4 eingenommen wird.

Überprüfen Sie die geänderte Funktionalität sowohl im Simulator als auch auf dem Praktikumsboard analog b) und c). Bestimmen Sie die hierfür benötigten Ressourcen.

Aufgabe 3

Es ist ein Design für ein Codeschloss zu entwickeln, welches folgende Forderungen erfüllt:

- Zu Beginn ist das Codeschloss bereit ⇒ RDY = '1'.
- Das Codeschloss soll nur nach Betätigung der Taster `BTN1`, `BTN2`, `BTN3`, `BTN4`, `BTN5` in dieser Reihenfolge öffnen. Alle anderen Tastenfolgen, mehrmaliges Betätigen derselben Taste hintereinander, als auch gleichzeitiges Drücken mehrerer Tasten stellen einen Fehlversuch dar.
- Bei richtiger Eingabe (und nicht gesperrtem Codeschloss, s.u.) wird der Ausgang OFFEN aktiv, bis ein Reset (`RES`) ausgelöst wird.
- Ein Fehlversuch kann nur durch den Korrekturschalter (`KOR`) oder durch einen Reset (`RES`) abgebrochen werden.
- Die durch den Korrekturschalter abgebrochenen Fehlversuche werden durch einen zweiten Automaten gezählt (Fehlerzähler).
- Nach dreimaliger Fehleingabe wird das Codeschloss gesperrt ⇒ RDY = '0'. Die Sperre verhindert ein Öffnen des Codeschlosses auch bei nachfolgender korrekter Eingabe des Codes. Die Sperre kann nur über einen Reset (`RES`) zurückgesetzt werden.
- Während der Korrekturschalter betätigt ist, sollen alle anderen Eingaben (Tasten außer Reset) ignoriert werden.
- Verwenden Sie für die Ein- und Ausgänge bitte folgende Signalnamen (wahlweise Groß-/Kleinschreibung) sowie Taster, Umschalter und LEDs auf dem Erweiterungsboard. Alle Ein- und Ausgänge sind high-aktiv.

Signal	Richtung	Bedeutung
CLK	Eingang	Board-Takt 1.842 MHz
BTN1	Eingang	Taster BTN1
BTN2	Eingang	Taster BTN2
BTN3	Eingang	Taster BTN3
BTN4	Eingang	Taster BTN4
BTN5	Eingang	Taster BTN5
RES	Eingang	Reset-Schalter SW1
KOR	Eingang	Korrekturschalter SW8
RDY	Ausgang	Bereit-Anzeige LED LD1
OFFEN	Ausgang	Schlossöffner LED LD2

- Alle weiteren LEDs LD3-LD8 können von Ihnen im Entwurf für Testzwecke z. B. zur Anzeige der Zustände verwendet werden.
- Tipp: Implementieren Sie die Automaten in separaten VHDL-Modulen, um eine saubere Trennung zu erhalten. So lässt sich die Funktionalität auch gut getrennt simulieren und testen.

a) Entwerfen Sie beide Automaten als SM-Chart. Die Automaten sind synchron miteinander zu verkoppeln.

b) Implementieren Sie die Automaten mittels VHDL und überprüfen Sie die korrekte Funktion im Simulator. Implementieren Sie dazu eine geeignete Testbench.

c) Erstellen Sie eine passende UCF-Datei für das Praktikumsboard. Synthetisieren Sie ihr VHDL-Modell und beheben Sie aufgetretene Fehler und Warnungen. Überprüfen Sie die korrekte Funktion des Codeschlosses auf dem Praktikumsboard.

Werten Sie im Praktikumsprotokoll die benötigten Ressourcen aus dem *Fit Report* aus.