Entwurf und Synthese von Eingebetteten Systemen

Sommersemester 2017

Übungsblatt 5

Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik
Lehrstuhl Eingebettete Systeme
Prof. Dr. O. Bringmann

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Montag, den 12. Juni 2017, 12:00

Die Abgabe der Lösungen erfolgt über das ILIAS-System als gepacktes Archiv (ZIP/TAR.GZ).
Dieses ZIP enthält die schriftlichen Antworten auf Fragen, die Quellcodes in Dateiform (nicht als Listing im PDF), und die Simulationsergebnisse als VCD-Dateien.

Abgabefrist: Montag, den 19. Juni 2017, 12:00 Uhr

Aufgabe 1: Fragen zur Vorlesung

[8 Punkte]

Bitte beantworten sie die nachfolgenden Fragen kurz!

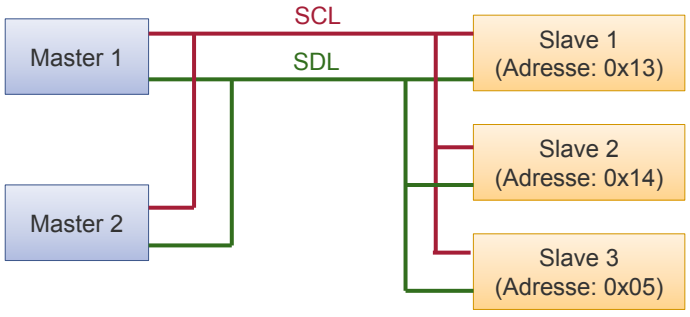
(a) Gegeben sei der nachfolgende Code. Würden bei einer Synthese dieses Prozesses Latch(es) inferiert werden? Begründen sie kurz!

```
1  process(SEL) begin
2    IF SEL = '1' THEN
3      Y <= ~Y;
4      X <= 0;
5    ELSE
6      X <= 1;
7    END IF;
8  END PROCESS
```

(b) Ein Bus A und ein Bus B werden über eine Bus-Bridge miteinander verknüpft. Die Teilnehmer am Bus B agieren hierbei ausschließlich als Slaves in der Kommunikation zwischen beiden Bussen und die Teilnehmer am Bus A ausschließlich als Master.

Weisen sie der Bridge jeweils eine Rolle (Bus-Master, Bus-Slave und Bus-Arbiter) zu, die es in Bus A und Bus B implementieren muss!

(c) Gegegeben sei das folgende I^2C-Bussystem mit zwei separaten Mastern und drei Slaves.



Folgende Daten sollen über den Bus übertragen werden:

- Master 1 möchte das Datum von Slave 1 ab SCL-Takt 1 lesen.
- Master 2 möchte das Datum 0x02 an Slave 3 ab SCL-Takt 1 übertragen.

Stellen sie tabellarisch die Daten auf der SDL-Leitung (pro SCL-Takt) sowie die Werte (0,1,Z,X,S,P) an den zwei Master- und zwei Slave-Ausgängen bis zum vollständigen Abschluss des ersten Transfers dar. Markieren (oder nennen) sie explizit die Stelle, an der das Arbitrierungsprotokoll eingreift.

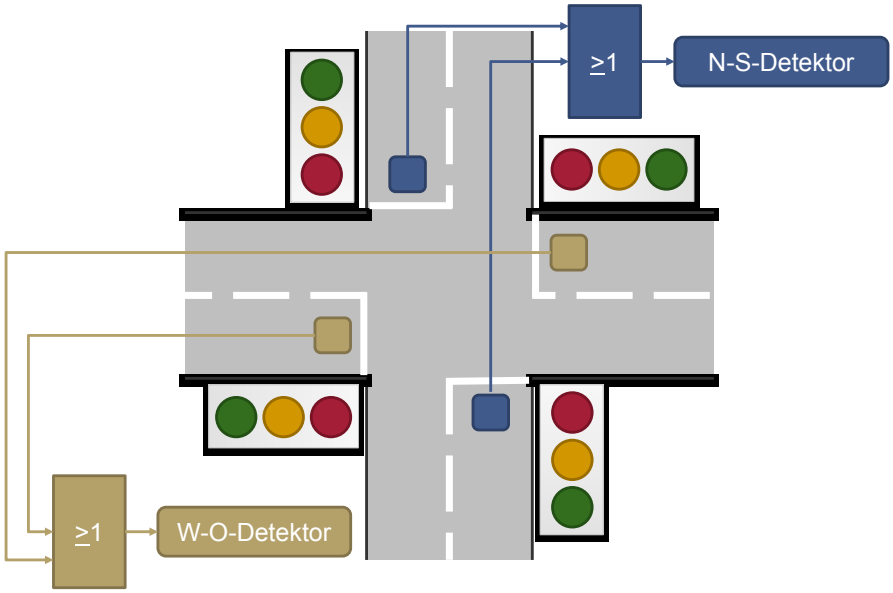
Beispieldarstellung mit fiktiven Werten:

Takt	1	2	3	...	...
SDL	S	0	1	...	P
M1	S	Z	Z	...	P
M2	S	0	Z	...	...
S1	Z	Z	Z	...	...
S3	0	0	1	...	...

Aufgabe 2: Modellierung einer Ampelsteuerung mit VHDL [12 Punkte]

Entwickeln Sie in VHDL eine Ampelsteuerung auf Basis eines expliziten Steuerwerks mit implizitem Datenpfad.

(a) Die Ampelsteuerung regelt die Ampeln für eine Kreuzung mit zwei gleichberechtigten Straßen – eine Straße in Nord-Süd-Richtung (N-S) sowie eine in West-Ost-Richtung (W-O). Die Ampeln der gleichen Straßenrichtung werden exakt gleich angesteuert. Vor den Ampeln befinden sich jeweils Schleifendetektoren, die im Boden eingelassen sind. Die Detektoren der gleichen Straßenrichtung sind jeweils über ODER-Gatter miteinander verbunden. Die nachfolgende Abbildung stellt die Ampelkreuzung inklusive der verbundenen Detektoren dar.



Die Ampelsteuerung soll die nachfolgende Schnittstelle implementieren:

Sensor- und Takteingaben:

```
clk, reset, ns_detektor, wo_detektor:  IN BIT
```

Ampelausgaben:

```
ns_ampel, wo_ampel:  OUT BIT_VECTOR(2 DOWNTO 0)
```

Die Ausgabevektoren stellen dabei die Ansteuerung der Leuchten dar:
Index 2 -> Rot, Index 1 -> Gelb, Index 0 -> Grün: jeweils high-aktiv

Funktionsbeschreibung:

Morgens beim Einschalten der Ampelanlage wird zuerst ein Reset (synchron, high-aktiv) durchgeführt, der beide Ampelpaare für 5s auf rot setzt. Anschließend wird das N-S Ampelpaar direkt auf grün (ohne Beachtung der Sequenz) geschaltet.

Jede Ampel durchläuft im Betrieb nach der Rotphase folgende Sequenz:
1. Schalten auf rot-gelb (für 5 Sekunden).
2. Umschalten auf grün.

Jede Ampel durchläuft im Betrieb nach der Grünphase folgende Sequenz:
1. Schalten auf gelb (für 5 Sekunden)
2. Umschalten auf rot.

Zwei Ampeln können niemals gleichzeitig eine Fahrbereitschaft (beide auf gelb/rot-gelb, beide auf grün) anzeigen und mit Ausnahme des Reset-Vorgangs dürfen beide Ampeln nie gleichzeitig auf rot stehen.

Das Umschalten von grün nach gelb und rot soll nur dann erfolgen, wenn der Schleifendetektor der kreuzenden Straße ein wartendes Fahrzeug anzeigt, d.h. erfolgt hier kein zeitbasiertes Umschalten.

Die Gelb-Phase sowie die Gelb-Rot-Phase soll jeweils 5 s dauern.

Nehmen Sie die Taktperiode als Zeitgeber für 5 s an.

Erwartete Abgaben: VHDL-Datei(en) für Ampelsteuerung

(b) Simulieren Sie die Ampelsteuerung mit Hilfe eines geeigneten Testtreibers, der bei beiden Ampel mindestens 2 Grünphasen hervorruft.

Erwartete Abgaben: VCD-Datei, Testtreiber (VHDL)

(c) Erweiterung Sie die Ampelsteuerung um einen **nebenläufigen Timer-Prozess**. Mit Hilfe dieses Timer-Prozesses soll die Ampelsteuerung so erweitert werden, dass die Grünphase für die Nord-Süd-Ampeln mindestens auf 30 Sekunden und die Grünphase der West-Ost-Ampeln mindestens auf 20 Sekunden verlängert. Erst nach Ablauf der jeweiligen Zeitspanne soll die Umschaltung erfolgen, wenn der Schleifendetektor der kreuzenden Straße ein wartendes Fahrzeug anzeigt. Nehmen Sie an, dass die Periode des Taktsignals 5 Sekunden dauert.

Die Kommunikation mit dem Zeitgeber-Prozess soll im Modus **Blockierendes Senden und Blockierendes Empfangen** geschehen. Der Beginn der Zeitmessung wird durch den Befehl `timeout_start` eingeleitet, gleichzeitig wird ein Signal `timeout_type` (Vektorbreite: 2 Bit) mitgegeben, welches die Ablaufzeit durch die Übertragung der Zehner-Dezimalstelle definiert, d.h. eine `2` für 20 Sekunden und eine `3` für 30 Sekunden. Der Start des Timers wird über ein `timeout_ack`-Signal bestätigt. Nach Ablauf der Zeitmessung setzt der Zeitgeber-Prozess das Signal `timeout` auf High.

Erwartete Abgaben: VHDL-Datei(en) für die erweiterte Ampelsteuerung

(d) Simulieren Sie die erweiterte Ampelsteuerung mit Hilfe eines geeigneten Testtreibers, der bei beiden Ampel mindestens 2 Grünphasen hervorruft.

Erwartete Abgaben: VCD-Datei, Testtreiber (VHDL)