# Entwurf und Synthese von Eingebetteten Systemen

## Sommersemester 2017

## Übungsblatt 4

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

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Montag, den 29. Mai 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 12. Juni 2017, 12:00 Uhr

## Aufgabe 1: Fragen zur Vorlesung

[9 Punkte]

Beantworten sie die nachfolgenden Fragen kurz!

(a) Gegeben sei der VHDL-Code zur Beschreibung eines getakteten Register mit einem asynchronen und high-aktiven Reset:

```
1  process(CLK, RST)
2  begin
3    if (RST = '1') then
4      Q <= '0';
5    elsif (CLK'event and CLK='1') then
6      Q <= D;
7    end if;
8  end process;
```

Schreiben sie den Code so um, dass sie ein getaktetes Register mit einem **synchronen** und **low-aktiven** Reset, d.h. Reset nur bei positiver Taktflanke wenn RST=0 gilt, beschreiben.

(b) Was versteht man unter Resource Sharing?

(c) Gegeben sei folgender VHDL-Code:

```
1  signal S: BIT_VECTOR(1 downto 0);
2  signal A: SIGNED(31 downto 0);
3  signal B: SIGNED(31 downto 0);
4  signal C: SIGNED(31 downto 0);
5  signal D: SIGNED(31 downto 0);
6  signal Z: SIGNED(31 downto 0);
7  p0: process(S,A,B,C,D) begin
8    case S is
9      when 00 =>
10       Z <= A + B;
11     when 10 =>
12       Z <= A * B;
13     when 01 =>
```

```
14          Z <= C + D;
15        when 11 =>
16          Z <= C * D;
17      end case;
18  end process;
```

Wieviele Addierer und Multiplizierer werden für dieses Verhalten benötigt, wenn kein Resource Sharing angewendet wird?

Wenden sie Resource Sharing an und schreiben sie den Code entsprechend um! Wieviele Addierer und Multiplizierer werden nun benötigt, wenn sie Resource Sharing für dieses Beispiel durchführen? (Der Code muss nicht kompilierbar sein!)

(d) Gegeben sei folgende Entity:

```
1  ENTITY my_ent IS
2    PORT( clk, A, B: IN STD_LOGIC;
3    X: OUT STD_LOGIC);
4  END my_ent;
```

Nachfolgend sind 3 Architectures dieser Entity dargestellt, welche die gleiche Zustandsmaschine inkl. Datenpfad modellieren, jedoch jeweils auf andere Art und Weise. Ordnen Sie jeder Architecture a) bis c) eine der Steuerwerksmodellierungstypen 1) - 4) zu.

**Architectures:**

**a)**

```
1  ARCHITECTURE my_arch_fsm1 OF my_ent IS BEGIN
2    Type state_type IS (ST0, ST1, ST2,...);
3    SIGNAL state : state_type := ST0; -- Zustandsregister
4
5    PROCESS(clk) BEGIN
6      IF (clk'EVENT AND clk = '1') THEN
7        CASE state IS
8          WHEN ST0 =>
9            X <= A and B;
10           state <= ST1;
11         WHEN ST1 =>
12           X <= A or B;
13           state <= ST2;
14         WHEN ST2 => ...
15       END CASE;
16     END IF;
17   END PROCESS;
18 END ARCHITECTURE
```

**b)**

```
1  ARCHITECTURE my_arch_fsm2 OF my_ent IS BEGIN
2    Type state_type IS (ST0, ST1, ST2,...);
3    SIGNAL state : state_type := ST0; -- Zustandsregister
4
5    PROCESS(state) BEGIN
6      CASE state IS
```

```
7         WHEN ST0 => X <= A and B;
8         WHEN ST1 => X <= A or B;
9         WHEN ST2 => ...
10      END CASE;
11    END PROCESS;
12
13    PROCESS(clk) BEGIN
14      IF (clk'EVENT AND clk = '1') THEN
15        CASE state IS
16          WHEN ST0 => state <= ST1;
17          WHEN ST1 => state <= ST2;
18          WHEN ST2 => ...
19        END CASE;
20      END IF;
21    END PROCESS;
22  END ARCHITECTURE
```

**c)**

```
1  ARCHITECTURE my_arch_fsm3 OF my_ent IS BEGIN
2    PROCESS BEGIN
3      WAIT UNTIL clk'event and clk='1'; --ST0
4      X <= A and B;
5      WAIT UNTIL clk'event and clk='1'; --ST1
6      X <= A or B;
7      WAIT UNTIL clk'event and clk='1'; --ST2 ...
8    END PROCESS;
9  END ARCHITECTURE
```

**Steuerwerksmodellierungsarten:**

**1)** Explizites Steuerwerk mit explizitem Datenpfad

**2)** Explizites Steuerwerk mit implizitem Datenpfad

**3)** Implizites Steuerwerk mit explizitem Datenpfad

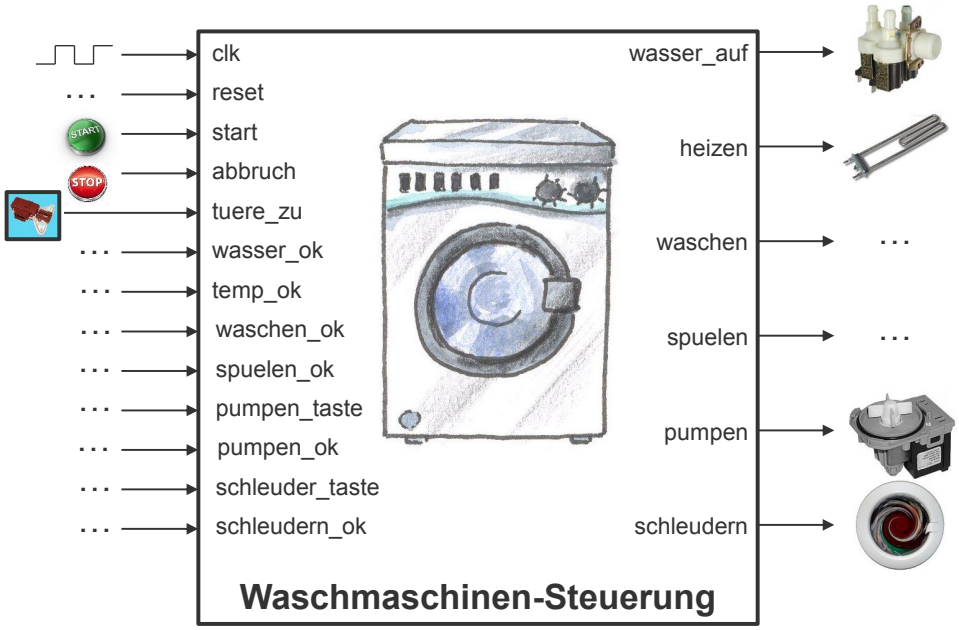
**4)** Implizites Steuerwerk mit implizitem Datenpfad

**Erwartete Abgaben:** Antworten im PDF

## Aufgabe 2: Waschmaschinensteuerung auf Basis einer expliziten FSM [11 Punkte]

(a) Beschreiben Sie in VHDL eine vereinfachte Waschmaschinensteuerung auf Basis einer expliziten FSM.

Die Waschmaschinensteuerung ist dabei zuständig für die Ansteuerung externen Komponenten (z.B. Pumpe: `pumpen`, Wasserzulauf: `wasser_auf`, ...), sowie für die Verarbeitung externer Eingaben von Sensoren (z.B. Türsensor: `tuere_zu`, ...) und Tasten (Start-Taste: `start`, ...) über eine I/O-Schnittstelle (siehe unten). Die nachfolgende Abbildung stellt die Waschmaschinensteuerung inkl. einiger angeschlossener externer Komponenten und Sensoren schematisch dar.




Die Waschmaschine soll die nachfolgende Schnittstelle implementieren:

Sensor-, Takt- und Tasteneingaben:

```
clk, reset, start, abbruch, tuere_zu, wasser_ok, temp_ok,
waschen_ok, spuelen_ok, pumpen_taste, pumpen_ok, schleuder_taste,
schleudern_ok:  IN BIT
```

Kommandos:

```
wasser_auf, heizen, waschen, spuelen, pumpen, schleudern:  OUT BIT
```

Alle Kommandos, Sensor- und Tasteneingaben seien high-aktiv. Der Reset ist high-aktiv und synchron.

**Erwartete Abgaben:** VHDL-Datei(en) der Waschmaschinensteuerung

(b) Zeichnen Sie ein Zustandsdiagramm der FSM im Waschmaschinen-Prozess.

**Erwartete Abgaben:** Diagramm im PDF

(c) Testen und simulieren Sie die Waschmaschine mit Hilfe eines geeigneten Test-Treibers.

**Erwartete Abgaben:** VHDL-Datei(en) des Testtreibers, VCD-Datei der Simulation mit Testtreiber

# Funktionsbeschreibung der Waschmaschinensteuerung:

Ein Standard-Waschvorgang führt sequentiell die folgenden Funktionen aus:

1. **Wasser einlassen:** Wasser wird in die Waschmaschine eingelassen.
2. **Heizen:** Das Wasser in der Waschmaschine wird erhitzt.
3. **Waschen:** Die Wäsche in der Waschmaschine wird gewaschen.
4. **Spülen:** Die Wäsche in der Waschmaschine wird gespült.
5. **Pumpen:** Das Wasser in der Waschmaschine wird abgepumpt bis sich kein Wasser mehr in der Waschmaschine befindet.
6. **Schleudern:** Die Wäsche in der Waschmaschine wird geschleudert.

Vor der Funktion *Wasser einlassen* und nach der Funktion *Schleudern* befindet sich die Waschmaschine im Ruhezustand.

Pumpen und Schleudern:
Die Waschmaschine soll nur dann pumpen, wenn die Taste `pumpen_taste` betätigt ist. Zusätzlich soll sie auch nur dann schleudern, wenn die Taste `schleuder_taste` betätigt ist. Schleudern kann man allerdings nur, wenn vorher das Wasser abgepumpt wurde.

Abbruchvorgang:
Ein Abbruch des Waschvorgangs (`abbruch`-Taste) soll aus jedem Zustand möglich sein. Nach dem Abbruch soll `pumpen_ok` (kein Wasser in der Waschtrommel) geprüft und eventuell das Wasser abgepumpt werden.

**Achtung:** Ein Reset (`reset`-Eingang) führt zum Zurücksetzen der Steuerung in den Initialzustand - hierbei wird nicht die Abbruchprozedur durchlaufen. Diese wird nur beim Drücken der `abbruch`-Taste durchlaufen.

Wasserzulauf und Heizen:
Der Wasserzulauf und die Heizung schalten zusätzlich zu den Sensorsignalen (`wasser_ok`, `temp_ok`) automatisch ab, wenn der Wasser-Füllstand bzw. die Temperatur erreicht sind. Trotzdem müssen alle Befehlssignale (z.B. `wasser_auf`, `heizen`) im Folgezustand zurückgesetzt werden.