Entwurf und Synthese von Eingebetteten Systemen

Sommersemester 2017

Übungsblatt 3

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

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Montag, den 22. 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.

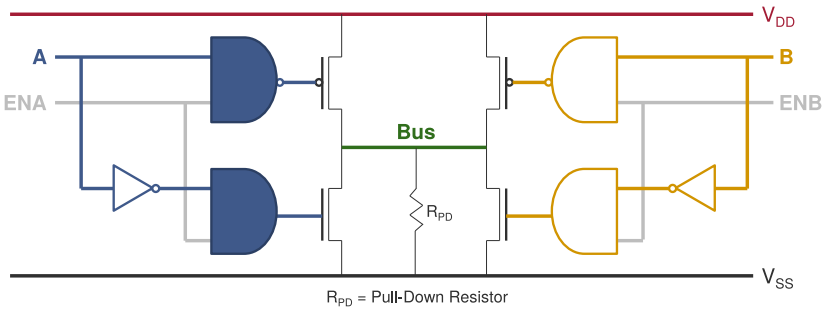
Abgabefrist: Montag, den 29. Mai 2017, 12:00 Uhr

Aufgabe 1: Fragen zur Vorlesung

[7 Punkte]

Beantworten sie die nachfolgenden Fragen kurz!

(a) Gegeben sei die folgende Schaltung mit zwei Tristate-Ausgängen, die auf denselben Bus schreiben.



Welche Signale liegen jeweils auf dem Bus für die nachfolgenden Fälle an? Begründen sie kurz!

(Fall 1) A=1, ENA=1, B=0, ENB=1
(Fall 2) A=1, ENA=0, B=1, ENB=0

(b) Was ist ein Deltazyklus? Wieviel Simulationszeit vergeht in einem Deltazyklus?

(c) Mit welchen Anweisungen kann man die Simulationszeit in VHDL beeinflussen?

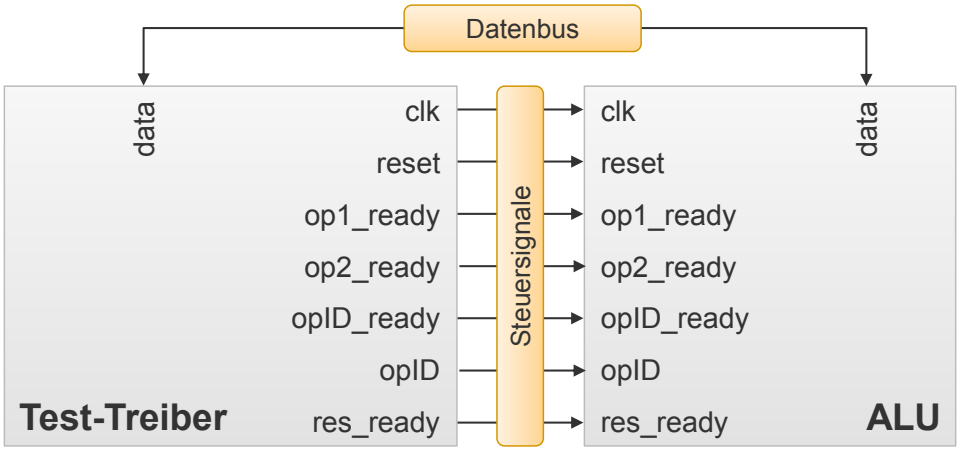
(d) Für das nachfolgende VHDL-Design: Wie viele Deltazyklen durchläuft die Simulation bis zum Voranschreiten der Simulationszeit, wenn sich der Eingang `a` ändert? Begründen sie!

```
1  ENTITY my_ent IS PORT(clk: IN STD_LOGIC,
2    a: IN STD_LOGIC;
3    z: OUT STD_LOGIC);
4  END my_ent;
5
6  ARCHITECTURE my_arch OF my_ent IS
7  BEGIN
8    PROCESS(a, z, clk) BEGIN
9      z <= z NAND a;
10   END PROCESS;
11 END my_arch;
```

Aufgabe 2: Tri-State-ALU mit Testtreiber in VHDL [13 Punkte]

In dieser Aufgabe sollen sie in VHDL eine ALU[1] mit einem integrierten Tri-State-Treiber beschreiben. Hierbei kann der Datenbus sowohl von einem externen Modul geschrieben und von der ALU gelesen werden (zur Übergabe der Operanden), aber auch von der ALU selbst geschrieben und vom externen Modul gelesen werden (zur Übergabe des Ergebnis).

(a) Beschreiben sie in VHDL eine einfache synchrone ALU mit integriertem Tri-State-Treiber für die Daten (`data`). Die ALU soll die Grundrechenarten Addition, Subtraktion und Multiplikation beherrschen. Das nachfolgende Schema stellt die zu implementierende Schnittstelle der ALU dar. Bitte achten sie bei Ihrer Implementierung auf die Einhaltung der Schnittstelle!



Die Signale der Schnittstelle ergeben sich wie folgt:

clk: IN BIT
Takt-Signal

rst: IN BIT
Reset-Signal

opID: IN BIT_VECTOR(1 DOWNTO 0)
Selektion des Operators
(00 => Addition, 01 oder 10 => Subtraktion, 11 => Multiplikation)

op1_ready: IN BIT
HIGH wenn Operand 1 auf Datenbus liegt

op2_ready: IN BIT
HIGH wenn Operand 2 auf Datenbus liegt

opID_ready: IN BIT
HIGH wenn opID-Eingang gültig ist

res_ready: OUT BIT
HIGH wenn gültiges Ergebnis auf Datenbus liegt

data: INOUT STD_LOGIC_VECTOR(31 DOWNTO 0)
Datenbus-Leitung (Tri-State)

Funktionsweise: Zu Beginn einer Operation wird ein `reset`-Signal erwartet, das alle Ausgaben auf `0` bzw. `Z` setzt. Bei `op1_ready` = `1` wird Operand 1 über den Daten-Bus eingelesen – entsprechend Operand 2 bei `op2_ready` = `1`. Bei `opID_ready` = `1` wird die Operations-ID (`opID`) eingelesen. Bei `res_ready` = `1` wird das entsprechende arithmetische Ergebnis auf den Datenbus ausgegeben.

[1] Arithmetic Logic Unit

Beispiel: Die Operation sei Addition. Bei `op1_ready = 1` wird über den Datenbus der erste 32-Bit-Operand eingelesen und in einer Variablen vom Typ SIGNED gespeichert. Entsprechend Operand 2 bei `op2_ready = 1` . Bei `opID_ready = 1` wird die Operations-ID = `00` eingelesen. Nach der Addition von Operand 1 und Operand 2 wird das Bit `res_ready = 1` gesetzt, das Ergebnis in den Typ `std_logic_vector` konvertiert und im selben Takt auf den Datenbus geschrieben.

Hinweis: Achten Sie darauf, dass eine Multiplikation zweier 32-bit-Werte zu einem 64-bit Wert führt. Um einen Fehler bzw. einen Abbruch der Simulation zu vermeiden, können Sie das Ergebnis über eine 64-bit Zwischenvariable auf 32-bit reduzieren:

```
VARIABLE ZWISCHEN: SIGNED(63 DOWNTO 0);
SIGNAL A, B, ZIEL: SIGNED(31 DOWNTO 0);
...
ZWISCHEN := A + B;
ZIEL <= ZWISCHEN(31 DOWNTO 0);
```

Erwartete Abgaben: VHDL-Datei

(b) Beschreiben sie eine Testbench mit integriertem Tri-State-Treiber, der über einen Daten-Bus und den Steuerleitungen mit der ALU verbunden ist. Der Test-Treiber schreibt erst die Operanden A und B über den Daten-Bus zusammen mit der gewünschten Operation in die ALU und liest dann das Ergebnis vom Datenbus, wenn die Steuerleitung `res_ready` von der ALU auf `1` gesetzt ist.

Der Test-Treiber prüft das Ergebnis automatisch mit der `ASSERT`-Anweisung.

Testen Sie mit Ganzzahl-(`INTEGER`)-Werten, die Sie vor/nach der Aus/Eingabe auf den/vom Bus entsprechend konvertieren. Testen Sie alle Operationen wenigstens zwei Mal.

Simulieren sie das System mit **MentorGraphics Questa Sim** und geben Sie sowohl den Testtreiber als auch das bei der Simulation generierte VCD-File ab.

Erwartete Abgaben: VHDL-Datei, VCD-Datei.