



Ausgabe: 20. November 2017

Abgaben {  Theorie entfällt
 Praxis 26. November 2017
Rücksprache 27./28. November 2017

Dies ist das **1. bewertete Aufgabenblatt**.

Beachten Sie den **Abgabetermin** im Kopf dieser Seite.

Die Gesamtpunktzahl, die Sie erreichen, wird direkt zu ihren Modulpunkten gerechnet.

Es werden **60 Punkte** benötigt, **um das Praktikum zu bestehen!**

Aufgabe 1: Taktuntersetzer (5 Punkte)

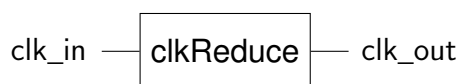


Abbildung 1: Entity clkReduce

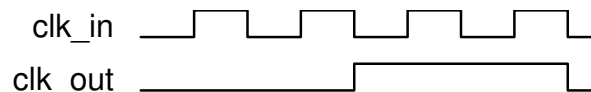


Abbildung 2: Taktuntersetzung mit
divisor = 4

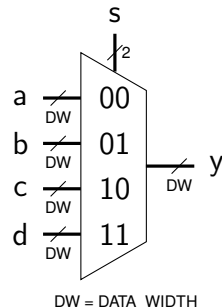
Name	Typ	Art	Beschreibung
divisor	integer	generic	Zählgrenze an der die Transition des Ausgangstaktes geschehen soll
clk_in	std_logic	in	Eingangstakt
clk_out	std_logic	out	Ausgangstakt

In dieser Aufgabe soll eine Taktuntersetzer für die Ansteuerung des 7-Segment-Displays entworfen werden.

Der Display arbeitet mit einer Wiederholungsrate von 1 kHz bis 10 kHz, unsere Zielplattform aber mit 100 MHz. Insofern muss eine Schaltung entwickelt werden, welche aus einem Eingangstakt `clk_in` mithilfe einer Zählgrenze `divisor` einen Ausgangstakt `clk_out` mit einer geringeren Frequenz generiert. Die Frequenz des Ausgangstaktes soll genau um den Faktor `divisor` geringer sein.

1. Implementieren Sie die `architecture behavioral` in der vorgegebenen Datei `clkReduce.vhd`. Nutzen Sie einen `process` zum Zählen.
2. Verifizieren Sie Ihr Design mithilfe der vorgegebenen Testbench, indem Sie im Aufgabenverzeichnis das Kommando `make clean all` ausführen.

Aufgabe 2: Mux4 (3 Punkte)



s	a	b	c	d	y
00	a	-	-	-	a
01	-	b	-	-	b
10	-	-	c	-	c
11	-	-	-	d	d

Abbildung 3: Entity mux4

Entwerfen Sie einen Multiplexer mit 4 Eingängen (a, b, c, d). Die Eingänge und der Ausgang haben eine variable Anzahl von Bits, welche durch den generic `DATA_WIDTH` bestimmt wird. Alle Eingänge sind `DATA_WIDTH` Bit breit und alle Bits sollen an den `DATA_WIDTH` Bit breiten Ausgang weitergeleitet werden.

Nutzen Sie zur Umsetzung ein nebenläufiges Statement!

Der Multiplexer soll eine Fehlerfortpflanzung erlauben (im Falle das sel nicht aus '0', '1' besteht)

1. Implementieren Sie Ihren Multiplexer in der `architecture behavioral`, welche Sie in der vorgegebenen Datei `mux4.vhd` finden.
2. Verifizieren Sie Ihr Design mithilfe der vorgegebenen Testbench, indem Sie in dem Aufgabenordner das Kommando `make clean all` ausführen.

Aufgabe 3: Hochladen auf Gitlab (2 Punkte)

1. (2 Punkte) Laden Sie die von Ihnen bearbeiteten Aufgaben in das Gitlab-Projekt Ihrer Gruppe hoch.
Nutzen Sie dafür die bereitgestellten Anleitungen auf *ISIS*. Das Gitlab-Tutorial beschreibt die Konfiguration von Gitlab und das Git-Tutorial erklärt den Umgang mit Git.

Literatur

- [1] Mentor Graphics Corporation. *ModelSim SE Reference Manual*, 6.4a edition.
- [2] Mentor Graphics Corporation. *ModelSim SE Tutorial*, 6.4a edition.
- [3] Mentor Graphics Corporation. *ModelSim SE User's Manual*, 6.4a edition.