



Ausgabe: 13. November 2023

Abgaben {  Theorie entfällt
 Praxis 19. November 2023
Rücksprache 20./21. November 2023

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

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

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

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

Aufgabe 1: Taktuntersetzer (5 Punkte)

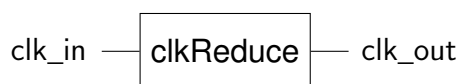


Abbildung 1: Entity clkReduce

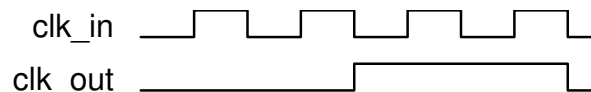


Abbildung 2: Taktuntersetzung mit
divisor = 4

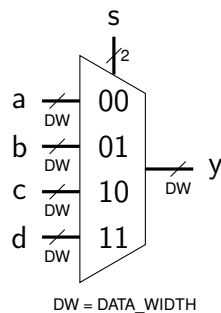
Name	Typ	Art	Beschreibung
divisor	integer	generic	Zählgrenze an der die Transition des Ausgangstaktes geschehen soll
clk_in	std_logic	in	Eingangstakt
clk_out	std_logic	out	Ausgangstakt

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: Generischer Mux4 (2 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 muss für einen ungültigen Wert von `sel`, der nicht nur aus '0', '1' besteht *keine* Fehlerfortpflanzung ermöglichen.

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: 7-Segment-Treiber (3 Punkte)

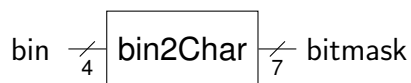


Abbildung 4: Entity bin2Char

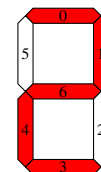


Abbildung 5: 7-Segment-Display

Name	Typ	in / out	Beschreibung
bin	<code>std_logic_vector(3 downto 0)</code>	in	binär kodierte Ziffer
bitmask	<code>std_logic_vector(6 downto 0)</code>	out	Steuerausgang für das 7-Segment Display. Eine '1' bringt die entsprechende LED im Display zum Leuchten.

Binär-Kodierung	Hexadezimal-Zeichen
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	b
1100	C
1101	d
1110	E
1111	F

In dieser Aufgabe soll ein `bin2Char`-Treiber für eine 7-Segment-Anzeige beschrieben werden, welche für jeden Eingangswert eine Bitmaske ausgibt. Diese Bitmaske wird zur Ansteuerung eines 7-Segment-Elements (vgl. 5) genutzt, sodass die hexadezimale Repräsentation des Eingangs auf dem Element angezeigt wird.

Für die Aufschlüsselung der Bitmaske (Bitmaskenposition zu LED-Position) betrachten Sie Abbildung 5.

1. Implementieren Sie die Funktionalität in der `architecture behavioral` der vorgegebenen Datei `bin2Char.vhd`. Legen Sie für das Mapping des Treibers von dem Eingang auf die Bitmaske ein `array` als `constant` an. Stellen Sie für die Implementierung zuerst eine Wertetabelle auf, in der Sie für jede Eingangskombination den Wert für die 7 Ausgangsbits festlegen. Diese Wertetabelle lässt sich dann direkt in das `constant array` übertragen.
2. Testen Sie Ihre Implementierung mit der vorgegebenen Testbench `bin2Char_tb` durch die Benutzung des Makefiles. Rufen Sie dazu im Aufgabenverzeichnis der Vorgaben das Kommando `make clean all` auf.

Literatur

- [1] ROrgPr Team. Rorgpr Übersicht. <https://rorgpr.gitlab-pages.tu-berlin.de/material/>.