Übung 6 Arithmetik, Arrays und Variablen

Jan Grapengeter

May 2, 2019

1 ieee.numeric_std.all Bibliothek

In der ieee.numeric_std.all Bibliothek sind mathematische Funktionen enthalten. Funktionen können nicht sequentiell sein. Funktionen können verwendet werden, um Code zu vereinfachen. Außerdem sind die Funktionen aus der Bibliothek bereits optimiert. Man sollte diese verwenden, anstatt den Code selber zu schreiben.

Aufgaben:
1.: Sehen Sie sich die Funktionen "+" und "*" in der ieee.numeric_std.all Bibliothek an und vergleichen Sie die "+" Funktion mit Ihrem Volladdierer.
2.: Implementieren Sie einen Multiplikator mit der Funktion "*" aus der ieee.numeric_std.all Bibliothek und mit denen Ihnen bekannten Methoden ohne Verwendung der Bibliothek. Sehen Sie sich die Schaltungen in "Elaborated Design" und "Synthesizer/Schematic" an und vergleichen Sie die Ergebnisse.

2 Arithmetische Funktionen

Die ieee.numeric_std.all Bibliothek enthält Funktionen zu Typumwandlung und arithmetischen Operationen auf Integern. Funktionen haben in VHDL immer einen Rückgabeparameter und können mehrere Eingabeparameter haben. Der Rückgabewert kann wie ein Signal gespeichert werden. Die Syntax für die Typumwandlung zwischen std_logic_vector und Unsigned ist:

Signal a: std_logic_vector (Länge downto 0);
Signal b: unsigned (Länge downto 0);
b<=unsigned(a);
a und b müssen die gleiche Länge haben. Falls dies nicht der Fall ist, kann man mit der Funktion "&" zwei Vektoren oder Unsigned zusammenfügen. Syntax:
Signal c: unsigned (Länge*2 downto 0);
c<=b&b;
Zur Erinnerung die Syntax der For-Schleife:
for I in Länge to 0 loop
SEQUENZ
if(BEDINGUNG) then
exit;
end if;
end loop;
Das exit Statement kann benutzt werden, um die Schleife vorzeitig zu beenden.

Aufgaben:
1.: Sehen Sie sich die Definition der "Shift_Left"-Funktion unter der folgenden Adresse an. Benutzen Sie dies, um eine Zahl mit zwei zu multiplizieren. Testen Sie Ihre Schaltung.
https://www.csee.umbc.edu/portal/help/VHDL/packages/numeric_std.vhd
2.: Erweitern Sie Ihre Schaltung zu einem kombinatorischen 8x8 Multiplizierer. Sehen Sie sich die Schaltung in "Elaborated Design" und in "Synthesizer/Schematic" an und vergleichen Sie die Ergebnisse mit denen Ihres ersten Multiplizierers.
3.: Überlegen Sie sich, wie Sie eine Shift-Funktion für std_logic_vector implementieren könnten.

3 Arrays

Mit Arrays lässt sich VHDL-Code vereinfachen. Die Syntax, um in VHDL ein Array anzulegen, ist:
type TYPENNAME is array of (LÄNGE) of ELEMENTTYP;

Aufgabe:
Verwenden Sie Ihr Wissen um Arrays und For-loops, um Ihren kombinatorischen Multiplizierer zu vereinfachen. Sehen Sie sich Ihre Schaltung in "Elaborated Design" und "Synthesis/Schematic" an und vergleichen Sie die Ergebnisse mit denen aus der vorherigen Aufgabe.

4 Variablen

Signale werden in FPGAs, wenn nicht anders angelegt, parallel verarbeitet. Wenn man VHDL in serielle Strukturen zwingen möchte, kann man Variablen verwenden. Variablen können nur innerhalb von Prozessen angelegt werden. Die Syntax dafür ist:
variable VARIABLENNAME : VARIABLENTYP;
Die Variablentypen sind die gleichen wie bisher bei Signalen.

Aufgaben:
1.: Implementieren Sie eine einfache Schaltung, mit der Sie die Unterschiede zwischen Signalen und Variablen demonstrieren können.
2.: Sehen Sie sich Ihre Schaltung in Elaborated Design" und "Synthesis/Schematic" an und vergleichen Sie die Ergebnisse.

5 Vergleichsfunktionen

Um die Größe von Vektoren/Integern zu vergleichen, können Vergleichsfunktionen benutzt werden. Für std_logic_vector sind die Vergleichsfunktionen standardmäßig definiert, für Integer wird die ieee.numeric_std.all Bibliothek verwendet.

Aufgaben:
1.: Implementieren Sie eine Schaltung, um die Vergleichsfunktion zu demonstrieren.
2.: Implementieren Sie eine eigene Vergleichsfunktion für Std_logic_vector.