Übung 3 Unterfunktionen, Signale und std_logic_vector

Jan Grapengeter

May 1, 2019

1 Halbaddierer

Die LUT eines Halbaddierers ist:

Halbaddierer LUT			
A1	A2	Übertrag	Summe
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 1: Halbaddierer LUT

Aufgabe: Implementieren Sie einen Halbaddierer in VHDL, legen Sie dafür ein Projekt an und verbinden Sie die Ein- und Ausgänge mit sw0, sw1, ld0 und ld1.

2 Volladdierer

Die LUT eines Volladdierers ist:

Volladdierer LUT				
A1	A2	cin	Übertrag	Summe
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 2: Volladdierer LUT

Aufgabe: Implementieren Sie einen Volladdierer in VHDL, erweitern Sie dazu das Projekt des Halbaddieres, nutzen Sie sw15 als cin.

3 Entity und Component

Um später komplexere Schaltungen bauen zu können, ohne den Code zu unübersichtlich werden zu lassen, können Teile von Schaltungen in Module ausgegegliedert werden, um diese später im

Hauptprogramm einbinden zu können. Dafür muss das Modul genau wie das Modul "main" definiert werden. Die Entity declaration muss im Code vor der architecture stehen, in der die Entity benutzt wird. Abfolge im Code:
1.: Libraries für "main"
2.: Entity declaration für "main" / Port map "main"
3.: Entity declaration für "subfunction(hier: volladdierer)" / Port map "subfunction"
4.: Architecture für "subfunction"
5.: Architecture für "main"
–> in 5. Component declaration
–> in 5. "begin"
–> in 5. Component instantiation
Die Syntax für Component declaration ist:
component KOMPONENTENNAME(hier: volladdierer) is
port(E1,E2,etc. : in std_logic;
A1,A2,etc. : out std_logic);
end component;

Die Syntax für Component instantiation ist:
INSTANZNAME(hier:voll1):KOMPONENTENNAME(hier:volladdierer)
port map(PORTNAME1 aus Component instantiation=>SIGNALNAME1 aus main,
wiederholen für alle Ports);
end component;
Damit wird ein Modul INSTANZNAME vom Typ KOMPONENTENNAME angelegt und dessen Ein- und Ausgänge mit Signalen der Haupfunktion verbunden.

Aufgabe:
Legen Sie ein neues Projekt in Vivado an und kopieren Sie Ihren Code für den Volladdierer aus der vorherigen Aufgabe. Legen Sie eine Entity und eine Component declaration an. Verbinden Sie die Ein- und Ausgänge der Komponente mit den gleichen Ein- und Ausgängen wie in der vorherigen Aufgabe.

4 Dateiausgliederung

Um den Code übersichtlicher zu halten, können Untermodule in eigene Dateien ausgegegliedert werden.
-Legen Sie ein neues Projekt an.
-Im Projektmanager rechtsklicken Sie auf "Design Sources" und wählen Sie "Add Source".
-Wählen Sie "Add or create design sources".
-W´ahlen Sie "Create File".
Wählen SIe VHDL als Dateityp, Volladdierer als Dateinamen und "Local to project" als Speicherort.
-"Finish"
-Legen Sie die Ein- und Ausgänge wie in der Entity declaration in der vorherigen Aufgabe fest.
-Wechseln Sie im Project Manager auf "Compile Order"
-Rechtklicken Sie auf Ihre neu angelegte Datei und wählen Sie "Move to Top"
Sie haben jetzt eine Datei zu Ihrem Projekt hinzugefügt, die vor ihrem Hauptprogramm kompiliert wird.

Aufgaben:
1.: Kopieren Sie Ihren Code der Entity declaration und der Architecture aus der vorherigen Aufgabe in die neu angelegte Datei.

2.: Legen Sie, wie in der vorherigen Aufgabe, eine Component Instantiation in main und verbinden Sie die Ein- und Ausgänge wie in der vorherigen Aufgabe.
3.: Kompilieren Sie Ihr Programm und schreiben Sie es auf das Basys3-board. Wenn alles funktioniert hat, sollte es das gleiche Verhalten wie in der vorherigen Aufgabe zeigen.
4.: Überlegen Sie sich, warum die Compile Order geändert werden musste.
5.: Legen Sie eine zweite Component Instantiation Ihres Volladdieres an und verbinden Sie diese mit beliebigen unbenutzen Ports.
6.: Sehen Sie sich Ihre Schaltung mit "Elaborated Design" und "Synthesis/Schematic" an.

5 Signale

Signale sind interne Repräsentationen von Drähten und internen elektrischen Signalen. Sie werden in VHDL wie folgt angelegt:
signal SIGNALNAME1,SIGNALNAME2,etc. : SIGNALTYP;
Diese Deklaration muss in der Architecture vor "begin" stehen.
Beispiel:
signal carry1,carry2 : std_logic;
Signale können wie Outputports gesteuert werden.
signal1 <= signal2;
signal3 <= inputport;
outputport <= signal4;
Signalen kann ein Anfangswert mitgegeben werden:
signal1 <= '1';
singal2 <= '0';

Aufgaben:
1.: Legen Sie Signale für alle Ihre Ein- und Ausgänge aus Ihrem Volladdierer aus der vorherigen Aufgabe an und verbinden Sie diese so, dass die Schaltung danach die gleiche Funktion hat wie zuvor.
2.: Optional: Sollten Sie Ihren Volladdierer bisher über die dysjunkte Normalform beschrieben haben, überlegen Sie sich, wie Sie den Code mit Signalen abkürzen können. Kompilieren Sie Ihre neue Schaltung und testen Sie diese.
3.: Sehen Sie sich Ihre Schaltung mit "Elaborated Design" und "Synthesis/Schematic" an.

6 Verbindung von Komponenten

Das Blockschaltbild eines 4-bit Volladdierers ist:

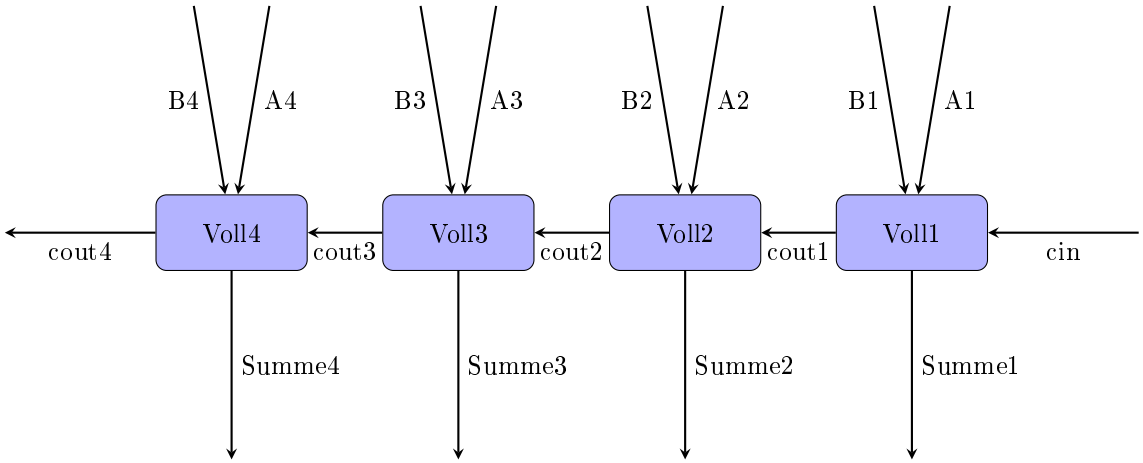

Figure 1: 4-bit Volladdierer

Aufgaben:
1.: Legen Sie ein neues Projekt an und implementieren Sie dort einen 4-bit Volladdierer. Kompilieren Sie Ihr Programm und testen Sie es auf dem Basys3-board.
2.: Sehen Sie sich Ihre Schaltung mit "Elaborated Design" und "Synthesis/Schematic" an.
3.: Welche Ein- und Ausgänge entsprechen welcher Zahl im Dezimalsystem?
4.: Erweitern Sie Ihre Schaltung auf alle Schalter und LEDs, die auf dem Basys3-board zu Verfügung stehen. Wie groß ist Ihr Volladdierer dann? Was ist das limitierende Kriterium?

7 std_logic_vector

Wenn Sie viele Ein- und Ausgänge ansteuern, wird die Port Map Ihrer Schaltung sehr lang. Diese kann jedoch verkürzt werden, indem gleiche Signaltypen in einen Vektor zusammengefasst werden. Die Syntax in der Port Map dafür ist:
VEKTORNAME : in/out std_logic_vector(VEKTORLÄNGE-1 downto 0);
Syntax, wenn Vektor als "signal" angelegt wird:
signal VEKTORNAME : std_logic_vector(VEKTORLÄNGE-1 downto 0);
Downto(VEKTORLÄNGE-1 downto 0) legt die Länge des Vektors auf VEKTORLÄNGE fest und definiert die nullte Stelle als LSB. Vivado benötigt für Vektoren eine andere Syntax in der Constraints Datei. Beispiel:
set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33 } [get_ports sw[0]];

Aufgaben:
1.: Legen Sie die Ein- und Ausgänge Ihres 4-bit Volladdierers als Vektor an und ändern Sie die Constraints Datei entsprechend.
2.: Legen Sie die Signale für ihre Komponenten als Vektoren an.
3.: Testen Sie Ihre Schaltung.
4.: Sehen Sie sich Ihre Schaltung mit "Elaborated Design" und "Synthesis/Schematic" an.

8 Generate Statement

Größere Volladdierer lassen sich nur umständlich per Hand anlegen. Dies kann jedoch über das Generate Statement gelöst werden. Damit lassen sich in einer Schleife eine definierte Anzahl von Komponenten instanzieren. Die Syntax dafür ist:

LABEL:
for i in 0 to ANZAHLKOMPONENTEN+1 generate
LABELNAME : KOMPONENTENNAME port map
(Komponentenport1=>signal1(i);Komponentenport2=>signal2(i),etc.);
end generate LABEL;
signal1 und signal2 müssen als Vektor angelegt sein, um in der Schleife zugeordnet werden zu können.

Aufgaben:
1.: Ändern Sie Ihren 4-bit Volladdierer so, dass die Komponenteninstanzierung nun über ein Generate Statement erfolgt. Ändern Sie Ihre Signale falls nötig.
2.: Sehen Sie sich Ihre Schaltung mit "Elaborated Design" und "Synthesis/Schematic" an.