Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Lehrstuhl für Technische Informatik
Prof. Dr. Manfred Schimmler
Dipl.-Inf. Lars Wienbrandt

FPGA-Entwurf mit VHDL

Sommersemester 2011

Serie 7

Abgabe: bis Fr. 17.06. um 10 Uhr im Schrein oder per Mail an lwi@informatik.uni-kiel.de.

Aufgabe 1

Erklären Sie anhand der Abbildung einer Spartan3-Slice aus dem Datenblatt zu Spartan3-FPGAs (*Fig. 5-2: Left-Hand SLICEM, p.200, ug331*) ausführlich wie eine einfache Ripple-Carry-Addition zweier 2-Bit-Zahlen in eine solche Slice gemappt wird. Die Ausgabe soll nicht in Register gespeichert werden.

Listen Sie die verwendeten Bauteile auf, beschreiben Sie, wie Sie miteinander verbunden sind und welche logischen Funktionen in die LUTs implementiert sind.

Hinweis: Die in blau gestrichelt dargestellten Bauteile und Verbindungen werden nicht benötigt und brauchen von Ihnen nicht betrachtet zu werden.

15 Punkte

Aufgabe 2 – Programmieraufgabe

Wie kann man auf einem Spartan3-FPGA ein Quad-Port-RAM implementieren, d.h. auf einem RAM-Block sollen vier Ports verfügbar sein, die alle auf den gleichen Speicherbereich zugreifen? Ein Port soll dabei die gleichen Signale zur Verfügung stellen und die gleiche Funktionalität besitzen, wie die Instanz eines Xilinx-Block-RAM-Primitives. Einzige Änderung: Alle Ports sollen mit dem gleichen Systemtakt betrieben werden.

a)

Implementieren Sie ein solches Quad-Port-RAM in VHDL für einen Spartan3-FPGA mit Wortbreite $9bit$ und Tiefe $2K$. Geben Sie auch evtl. neu entstandene Voraussetzungen an. Es ist Ihnen auch erlaubt die Schnittstelle Ihres RAMs geringfügig zu erweitern, wenn dadurch z.B. bei Verwendung mehrerer Instanzen Ressourcen eingespart werden können. Sie müssen nicht die verschiedenen Generics eines Xilinx-Block-RAM-Primitives implementieren.

b)

Testen Sie Ihre Implementierung.

15 Punkte

Aufgabe 3 – Programmieraufgabe

Implementieren Sie eine Laufschrift für das 7-Segment-Display des Memec-Spartan3-Boards.

a) Die Laufschrift soll nacheinander alle Ziffern einer Hexadezimalzahl generischer Größe für Menschen lesbar anzeigen. Beginnen Sie mit der höchstwertigen Ziffer. Führende Nullen sollen dabei nicht angezeigt

werden. (Die Null selbst soll natürlich trotzdem dargestellt werden.) Per Default zeigt das Display nichts an.

Hinweis: Zwischen zwei Ziffern sollten Sie eine kurze, aber sichtbare Pause (ohne Anzeige) einfügen, da sonst zwei gleiche aufeinanderfolgende Ziffern schlecht unterscheidbar wären. Nach der letzten Ziffer fügen Sie bitte erneut eine Pause der gleichen Länge, wie die Anzeigedauer einer Ziffer, ein.

Implementieren Sie folgende Schnittstelle:

```
entity ticker is
generic (
  NUMBER_WIDTH : positive := 32
);
port (
  clk : in std_logic;
  sync_reset : in std_logic;
  number_in : in std_logic_vector(NUMBER_WIDTH-1 downto 0);
  new_number_in : in std_logic;
  finished_out : out std_logic;
  display_out : out std_logic_vector(6 downto 0)
);
end ticker;
```

clk: Taktsignal

sync_reset: Taktsynchroner Rücksetzeingang. Bei "1" soll der Default-Zustand wieder hergestellt werden.

number_in: Die anzuzeigende Hexadezimalzahl.

new_number_in: Bei "1" beginnt ihre Einheit bei der nächsten positiven Taktflanke mit der Anzeige der Zahl. Es kann also angenommen werden, dass dieses Signal nur für eine Taktperiode lang gesetzt ist. Wird dieses Signal während einer laufenden Anzeige erneut gesetzt, beginnt die Anzeige aufs Neue. Das Eingangssignal `number_in` ist auch nur dann valide, wenn `new_number_in` auf "1" ist.

finished_out: Bei "0" läuft die Anzeige. Nach der letzten Pause wechselt dieses Signal auf "1". Der Default ist ebenfalls "1".

display_out: Ansteuerung für das 7-Segment-Display. "1": Segment leuchtet; "0": Segment leuchtet nicht.

Hinweis: Benutzen Sie die VHDL-Einheit `conv_7seg` aus dem auf der Homepage zur Verfügung gestelltem Projekt für die Anzeige auf dem Display.

b)

Testen Sie Ihre Einheit. Natürlich dürfen Sie für den Test die Anzeigedauer und Pausenlänge verringern.

c)

Binden Sie Ihre Einheit in das auf der Homepage zur Verfügung gestellte ISE-Projekt ein. Sie ermöglicht das Testen Ihres Designs auf dem Memec-FPGA-Board. Generieren Sie eine Konfigurationsdatei.

20 Punkte

Zusatzaufgabe 4 – Programmieraufgabe

Hexadezimalziffern sind für Menschen im allgemeinen schwer zu interpretieren. Implementieren Sie die gleiche Einheit aus Aufgabe 3, nur soll die eingehende Zahl nun dezimal ausgegeben werden. Auch hier sollen führende Nullen nicht ausgegeben werden. Testen Sie Ihr Design und generieren Sie eine Konfigurationsdatei.

Hinweis: Die Schwierigkeit dieser Aufgabe besteht eigentlich nur darin, die eingehende Zahl in ein Format zu konvertieren, in dem die einzelnen Ziffern nicht größer als "9" sind. Ihr Design darf dafür auch mehrere Taktzyklen verwenden.

10 Zusatzpunkte

Die Abgabe der Programmieraufgaben soll bitte folgendermaßen erfolgen:

1. **Verschieben Sie oder benennen Sie eine evtl. generierte Konfigurationsdatei zunächst um, damit sie beim Cleanup nicht gelöscht wird!**
2. Dann führen Sie in ISE "`Project` → `Cleanup Project Files`" aus.
3. Verpacken Sie den Projektordner in ein gepacktes Archiv (`.zip`, `.tar.gz` o.ä.).
4. Senden Sie das Archiv per Mail an `lwi@informatik.uni-kiel.de`