2 Projekt-Aufgabe 1

2.1 Ergänzung des Entwurfs eines 2-Bit-Zählers

In dem ersten Teil der Projektaufgabe sollen Sie den VHDL-Source-Code eines 2-Bit-Zählers ergänzen und simulieren (siehe Abbildung 1 und VHDL-Vorlage auf Moodle twobitcounter.vhd).

Der 2-Bit-Zähler soll wie folgt funktionieren:

- Realisierung eins Zwei-Bit-Zählers von 0..3 (Ausgabe des Zählerstandes an q_0 und q_1
- Es soll hoch- und heruntergezählt werden können (Eingang up = 1: Zähler soll hochzählen; Eingang down = 1: Zähler soll herunterzählen).
- Ein Carry- und ein Borrow-Bit sollen einen Überlauf- bzw. Unterlauf darstellen.
- Als Ergänzung zu dieser kurzen Spezifikation sei das korrekte Simulationsergebnis des Zählers in Abbildung 2 dargestellt.

2.1.1 Detaillierte Aufgabenstellung

- Ergänzen Sie den bereits vorhandenen VHDL-Source-Code des 2-Bit-Zählers derart, dass dieser die gegebene Spezifikation erfüllt.
- Erstellen Sie für die Durchführung der Simulation ein Do-File
- Weisen Sie die korrekte Funktionsweise Ihres Zählers durch Simulation nach.

2.1.2 Geforderte Ergebnisse/Dokumentation

Alle Ergebnisse sollen in Moodle in digitaler Form[2] abgegeben werden:

- Der von Ihnen vollständig ergänzte VHDL-Source-Code des Zählers. Bitte kennzeichnen Sie den VHDL-Code durch Ihren Namen, Datum und Matrikelnummer.
- Das von Ihnen erstellte Do-File für die Simulation. Auch hier bitte darauf achten, dass Sie die Datei durch Ihren Namen, Datum und Matrikelnummer ergänzen.
- Ein Screenshot der Simulation.

[2] Digitale Form bedeutet alles in einer gepackten Dateien (z.B. ZIP, RAR) in Moodle hochzuladen. Beachten Sie bitte das Abgabedatum, danach ist keine Abgabe mehr möglich, und somit natürlich auch keine Bewertung mehr möglich.

```
-------------------------------------------------
--  EXAMPLE OF TWO BIT COUNTER  --
-------------------------------------------------
-- Library Declaration --
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

-----------------------------------------
--      ENTITY                         --
-----------------------------------------
ENTITY twobitcounter IS
PORT(
     reset    :IN  std_logic;
     clk      :IN  std_logic;
     up       :IN  std_logic;
     down     :IN  std_logic;
     q_0      :OUT std_logic;
     q_1      :OUT std_logic;
     b        :OUT std_logic; --borrow
     c        :OUT std_logic  --carry
     );
END twobitcounter;

-----------------------------------------
--        ARCHITECTURE                 --
-----------------------------------------
ARCHITECTURE behave OF twobitcounter IS

TYPE state IS (st_0,st_1,st_2,st_3);
SIGNAL mode, nxt_mode : state;

BEGIN

-- Registered Process --
clk_proc : PROCESS (clk,reset)
     BEGIN
          IF (reset='0') THEN -- Active Low Reset --
               mode <= st_0;
          ELSIF (clk'EVENT AND clk='1' AND clk'LAST_VALUE='0') THEN
               mode <= nxt_mode;
          END IF;
     END PROCESS clk_proc;

-- Combinational Process --
counter_proc : PROCESS (mode,up,down)
     BEGIN
     CASE mode IS
          WHEN st_0 =>
               IF up='1' AND down='0' THEN
                    nxt_mode <= st_1;
               ELSIF up='0' AND down='1' THEN
                    nxt_mode <= st_3;
               END IF;
          WHEN st_1 =>
     --- Complete the statements !! ---
               END IF;
          WHEN st_2 =>
     --- Complete the statements !! ---
               END IF;
          WHEN st_3 =>
     --- Complete the statements !! ---
     END CASE;
END PROCESS counter_proc;

-- Output Process --
output_proc : PROCESS (mode,up,down)
     BEGIN
          IF mode=st_0 AND up='0' AND down='1' THEN
               b<='1';
          ELSE b<= --- Complete the statement !! ---
          END IF;
--- Complete the statement !! ---
          IF mode=st_3 AND up='1' AND down='0' THEN
               c<= --- Complete the statement !! ---
--- Complete the statement !! ---
                 IF mode=st_0 OR mode=st_1 THEN
               q_1<= --- Complete the statement !! ---
--- Complete the statement !! ---
          IF mode=st_0 OR mode=st_2 THEN
               q_0<= --- Complete the statement !! ---
     END --- Complete the statement !! ---
--- Complete the statement !! ---
```

Abbildung 1: Vorhandener VHDL-Source-Code des 2-Bit-Counters

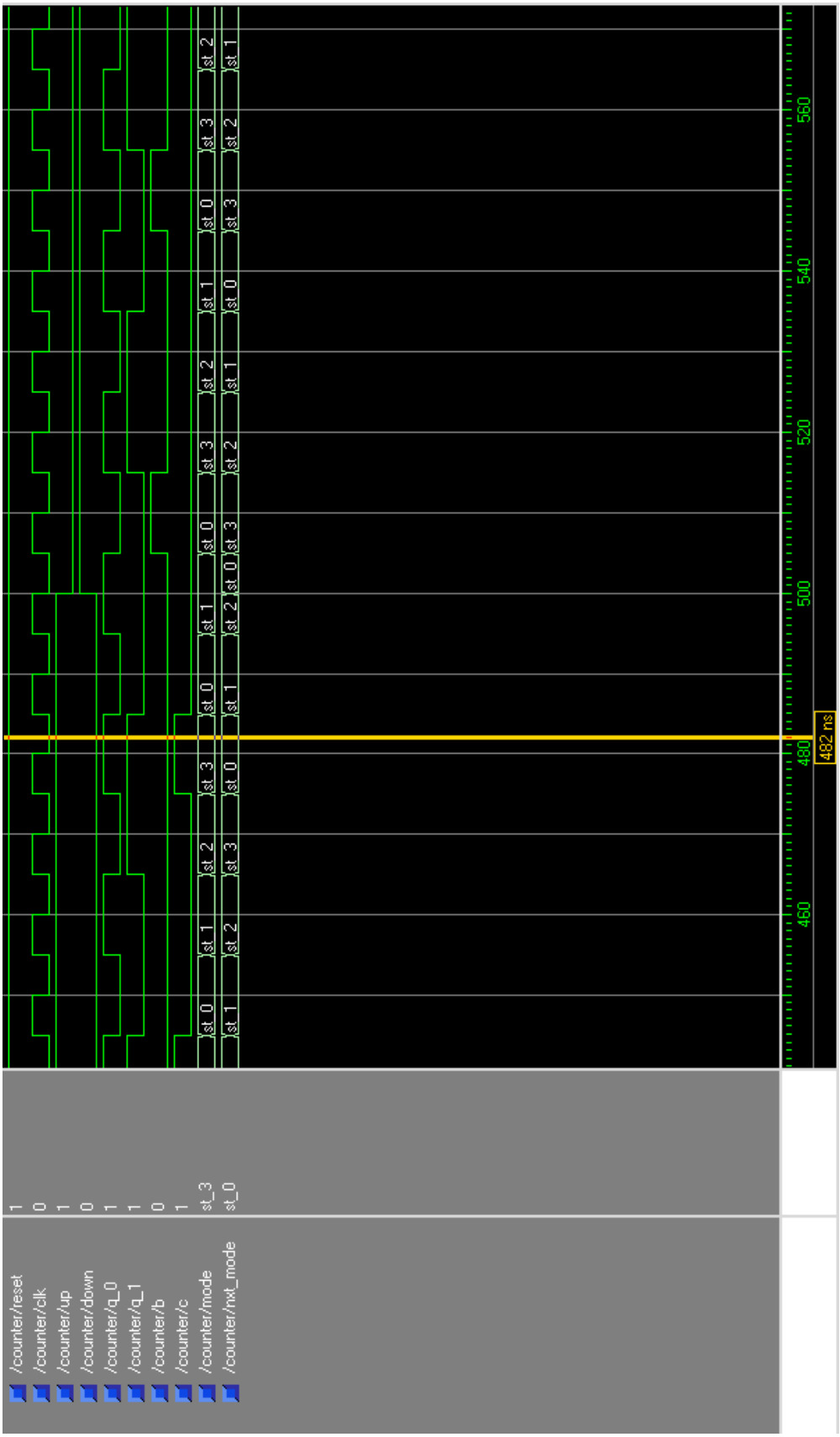

Abbildung 2: Korrekte Simulation des 2-Bit-Zählers

2.2 Entwurf eines BCD-Zählers mit 7-Segment-Anzeige

In diesem Aufgabenteil sollen Sie einen BCD-Zähler mit 7-Segment-Anzeige entwerfen. Die korrekte Funktionsweise/Realisierung des Zählers soll auf dem DE2-115-Development-Board getestet und nachgewiesen werden

2.2.1 Spezifikation des BCD-Zählers mit 7-Segement-Anzeige

- Es soll ein 4-Bit-BCD-Zähler von 0 bis 9 mit Darstellung des Zählerstandes auf einer 7-Segmentanzeige realisiert werden. Das prinzipielle Blockschaltbild des BCD-Zählers mit 7-Segment-Anzeige ist in Abbildung 3 gegeben.
- Der Zähler soll über zwei Eingänge Reset und Enable verfügen. Reset und Enable sollen je durch einen der DIP-Switches auf dem DE2-115-Development-Board realisiert werden.
- Der BCD-Zählers soll immer um eins hochzählen, falls der Enable-Eingang auf 1 gesetzt ist. Falls der Enable-Eingang auf 0 gesetzt ist, soll der aktuelle Zählerstand am Ausgang des BCD-Zählers stehenbleiben.
- Bei Anlegen der Versorgungsspannung soll der Zähler bei 0 anfangen zu zählen, falls der Enable-Eingang auf 1 gesetzt ist.
- Sobald der Reset-Taster betätigt wird, soll der Zähler wieder bei 0 angefangen zu zählen.
- Nach Erreichen der 9, soll der Zähler wieder bei 0 angefangen zu zählen.
- Die Ausgabe des Zählerstandes soll auf einer der 7-Segement-Anzeigen des DE2-115-Development-Boards erfolgen.
- Für die Realisierung des Taktes soll der auf dem DE2-115-Development-Board vorhandene Takt verwendet werden. Das Hochzählen des Zählers soll jedoch mit einer Frequenz von 1 Hz erfolgen.

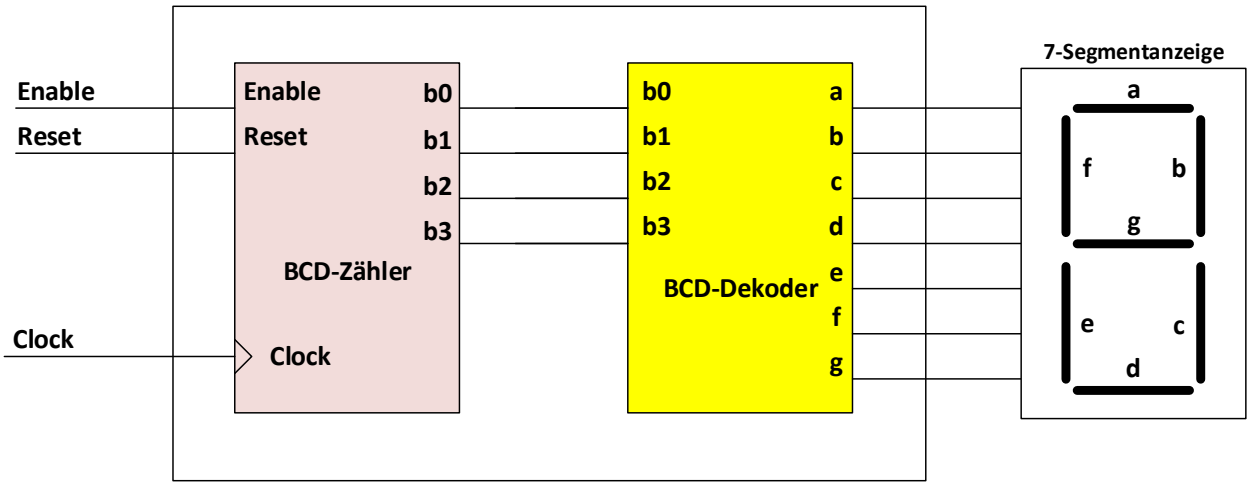

Abbildung 3: Prinzipielles Blockschaltbild des BCD-Zählers mit 7-Segment-Anzeige

2.2.2 Detaillierte Aufgabenstellung

Es ist ein hierarchisch aufgebauter, synthetisierbarer VHDL-Entwurf zu erstellen.

- Erstellen Sie zunächst ein Konzept/Design für Ihre Schaltung. An Hand des Designs soll der spätere VHDL-Entwurf nachvollziehbar sein. Ihr Design kann alle Designmethoden für den Schaltungsentwurf enthalten. Z.B. der Entwurf eines Blockschaltbilds, aus dem die Teilkomponenten des Systems hervorgehen, der Entwurf eines Zustandsdiagramms, der Entwurf von Booleschen Gleichungen, der Entwurf von Ablaufdiagrammen usw.
- Entwerfen Sie nach Erstellung des Designs Ihre VHDL-Beschreibungen. Achten Sie darauf, dass Ihr VHDL-Design ausreichend Kommentare enthält, wie dies bei der Programmierung notwendig ist.
- Simulieren Sie Ihren VHDL-Entwurf funktional mit Modelsim unter der Verwendung geeigneter Do-Files. Auch das Do-File sollte ausreichend Kommentare enthalten, so dass man das Do-File versteht.
- Synthetisieren Sie die Schaltung.
- Simulieren Sie die synthetisierte Schaltung auf Gatterebene mit Modelsim. Betrachten Sie sich den Unterschied zwischen der Simulation vor und nach der Synthese. Dokumentieren Sie den Unterschied.
- Überprüfen Sie die Funktion Ihrer Schaltung an der Hardware!
- Dokumentieren Sie Ihre Arbeit (maximal 15-20 Seiten)

2.2.3 Geforderte Ergebnisse/Dokumentation

Alle Ergebnisse sollen in Moodle in digitaler Form[3] abgegeben werden:

- Die Dokumentation der Aufgabe (Design, Simulationsergebnisse etc.) in einer einzigen pdf-Datei. Bitte kennzeichnen Sie die Dokumentation auf einem Deckblatt mit Ihrem Namen, Datum und Matrikelnummer.
- Der VHDL-Source-Code, des BCD-Zählers. Bitte kennzeichnen Sie den VHDL-Code durch Ihrem Namen, Datum und Matrikelnummer.
- Das von Ihnen erstellte Do-File für die Simulation. Auch hier bitte darauf achten, dass Sie die Datei durch Ihren Namen, Datum und Matrikelnummer ergänzen.
- Vorführung des Zählers an der Hardware.

[3] Digitale Form bedeutet alles in einer gepackten Dateien (z.B. ZIP, RAR) in Moodle hochzuladen. Beachten Sie bitte das Abgabedatum, danach ist keine Abgabe mehr möglich, und somit natürlich auch keine Bewertung mehr möglich.