Laborbericht 1

Entwurf eines BCD Zähler mit 7-Segment-Anzeige

Autor : Steve Wagner
Matrikelnummer : EI-3nat 175309
Professor : Prof. Dr.-Ing. Elke Mackensen

Datum : 10/11/2015

Steve Wagner
EI-3nat 175309

Hochschule Offenburg
University of Applied Sciences

Zusammenarbeit

1. Einführung 1
2. Architekturen 2
   2.1 Zustandsdiagramm 2
   2.2 Konzept der Schaltung 3
   2.3 VHDL-Architektur 4
3. Simulation 5
   3.1 Simulation vor Synthese 5
   3.2 Simulation nach Synthese 8
4. Konklusion 11

Steve Wagner
EI-3nat 175309

1. Einführung

In dieses Labor sollte einen BCD-Zähler mit der 7-Segment-Anzeige auf der DE2-115-Development-Board entwerfen.

Diese Zähler sollte mit einem VHDL-Programm erstellen, welches getestet und nachgewiesen sein wurde.

Dieses Labor hat mehrere Spezifikationen:

Zuerst, dieser BCD-Zähler soll von 0 bis 9 zahlen und diese Zahl auf einem 7-Segment anzeigen.

Nach Erreichen der 9, soll der Zähler wieder bei 0 anfangen.

Dieser Zähler soll auch über zwei Eingänge (Reset und Enable) abhängen.

Der Enable-Eingang soll auf 1 (mit einem Switch) gesetzt, um eins hochzuzählen. Andersfall soll der aktuelle Zählerstand stehenbleiben.

Sobald der Reset-Taster betätigt wird, soll der Zähler wieder bei 0 anfangen zu zählen.

Der Zähler soll mit einer Frequenz von 1 Hz hochzählen.

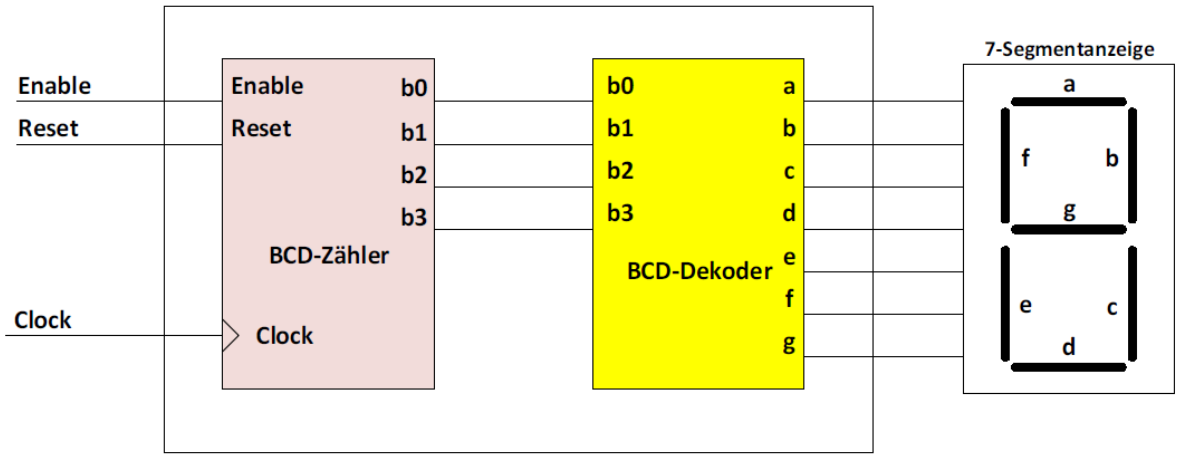

Abbildung 1 Prinzipielles Blockschaltbild des BCD-Zählers mit 7-Segment-Anzeige

2. Architekturen

2.1 Zustandsdiagramm

Um die Spezifikation zu beachten, habe ich ein Zustandsdiagramm erstellen.

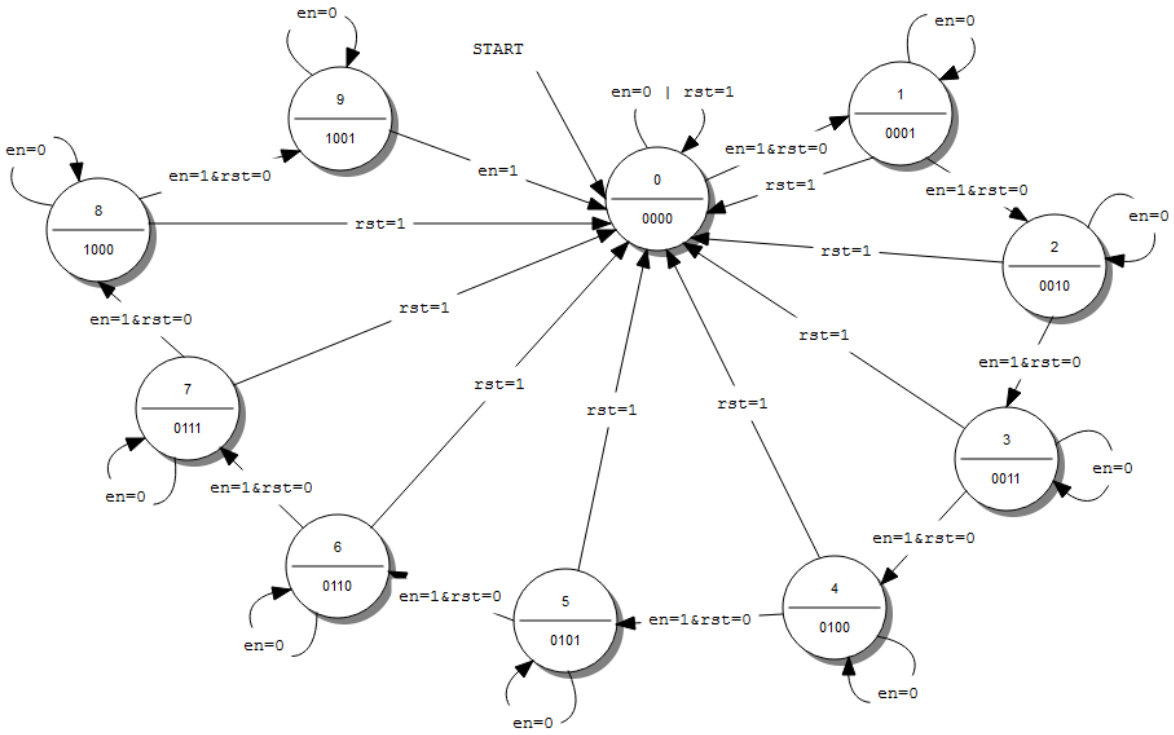

Abbildung 2 Zustandsdiagramm

Der Zähler fängt am 0 an und wenn der Enable-Switch zu 1 gesetzt ist, zählt es hoch.

Nachdem der Enable-Switch wieder zu 0 gesetzt ist, bleibt der Zähler in dem letzten Zustand.

Außerdem ist der Reset-Taster betätigt, fängt der Zähler wieder zu 0 an.

Wenn der Zahl 9 erreicht ist, dann fängt in die nächste Sekunde der Zähler wieder bei 0 an.

2.2 Konzept der Schaltung

Um der VHDL-Programm zu erstellen, habe ich diese Blockschaltbild gemacht. Mit diesem habe ich die Architektur des Programms herstellen.

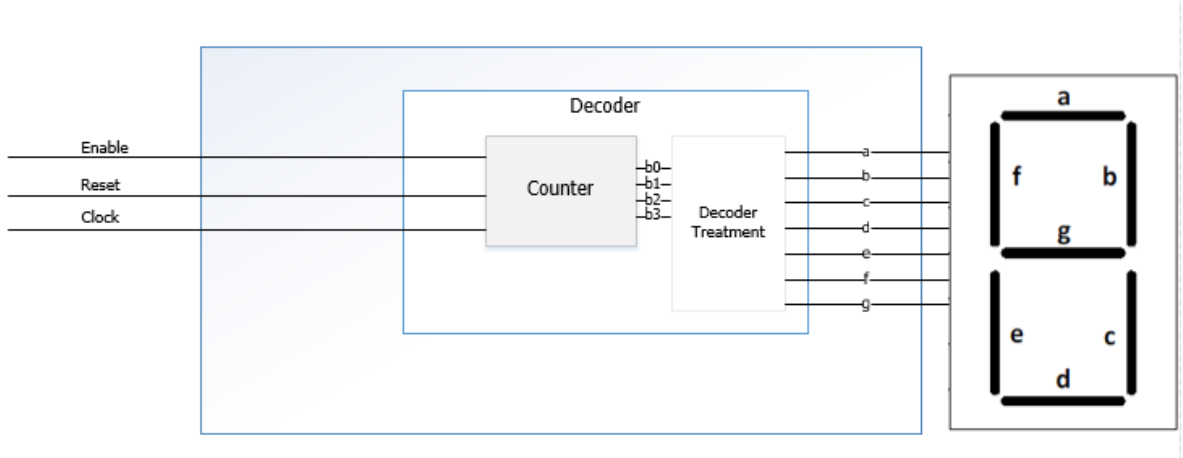

Abbildung 3 Blockschaltbild für VHDL Architektur

Es gibt 2 Hauptblöcke.

Der erste Block, Counter, ist für die Zähle von 0 bis 9 am bool'sche Ausgang zu erreichen. In diesem Block ist auch die 1-Hz-Frequenz hergestellt.

Der zweite Block, Decoder, hängt von den Werten der Counter-Ausgang (b0, b1, b2, b3) ab. Nachdem Werten des Counter-Ausgang, sind jede Segment des BCD ein- oder ausgeschaltet, um der Zahl der Counter anzuzeigen.

Die Abhängigkeit des 7-Segments von dem Decoder-Ausgang die auch von dem Counter-Ausgang abhängt, ist in die untere Wahrheitstabelle dargestellt. Um ein Segment einzuschalten, soll ein ‚0' gesetzt und um ein Segment auszuschalten, soll ein ‚1' auf dem Segment gesetzt

Zahl	b0	b1	b2	b3	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Abbildung 4 Wahrheitstabelle

2.3 VHDL-Architektur

Wie ist schon in der letzte Schritt geschrieben, hängt der VHDL Architektur nach dem Abbildung 3 ab.

Der Decoder Block ist der Hauptblock. In VHDL dieser Block ist den Top-Entity, weil es die Bindung zwischen der prinzipielle Eingang (Zahl) und der prinzipielle Ausgang (7-Segment) macht.

Der Counter Entity ist in der Decoder Architektur nach einem COMPONENT gezeichnet. Dann nach einen Port Map sind die richtige Signale in der Decoder Architektur verbinden.

Das erlaubt, um eine lesbare Projekt zu haben.

Um die reale Simulation mit der 1-Hz-Frequenz zu haben, ist die untere Zeile in den BcdCounter.vhd zu de-kommentieren.

Die Frequenz der Board ist 50MHz. Man soll also ein Counter von 50.000.000 haben, um 1Hz zu erreichen.

```
--              IF (countClk = 50000000) THEN --1 second passed
--                  countClk<=0;
                    countBcd<=countBcd+1;
                    IF (countBcd=9) THEN
                        countBcd<=0;
                    END IF;
--              END IF;
```

3. Simulation

Es gibt ein DoFile, bcdCounter.do, um die Simulation zu starten.

3.1 Simulation vor Synthese

In diesem Fall habe ich Modelsim geöffnet und dann die Projekt hochgeladen.

Für die Simulation habe ich der 1-Hz-Frequenz nicht benutzen, weil die Simulation zu lang war, um mehrere Sekunde zu simulieren.

Die Simulation vor Synthese ist nur die Interpretation der Kode

Zuerst habe ich den Counter getestet (Siehe Abbildung 5).

Man sieht, dass ab 100ns fängt der Counter an, weil der Enable-Eingang zu 1 gesetzt ist. Es zählt von 0 bis 9 hoch und fängt wieder bei 0 an wenn der Zahl 9 erreichen ist.

Ab 275ns gibt es ein Reset. Der Zähler fängt wieder bei 0 an.

Es gibt noch eine andere Reset ab 380ns, die lange zu 0 bleibt. Man sieht dann, dass der Zähler zählt wieder hoch nur wenn der Reset-Eingang wieder zu 1 geht.

Ab 410 ns geht der Enable-Eingang zu 0. Der Zähler ist gestoppt und bleibt an dem letzten Zustand (2 in diesen Fall).

Ab 450ns ist der Enable-Eingang wieder zu 1 gesetzt und zählt der Zähler wieder hoch.

Abbildung 6 ist die Darstellung von dem Decoder mit den gleichen Schritte als oben repräsentiert. Es gibt nur die Werten von dem 7-Segment (a, b, c, d, e, f, g) mehr als in Abbildung 5, die mit der Wahrheitstabelle der Abbildung 4 vergleichen sein könnte.

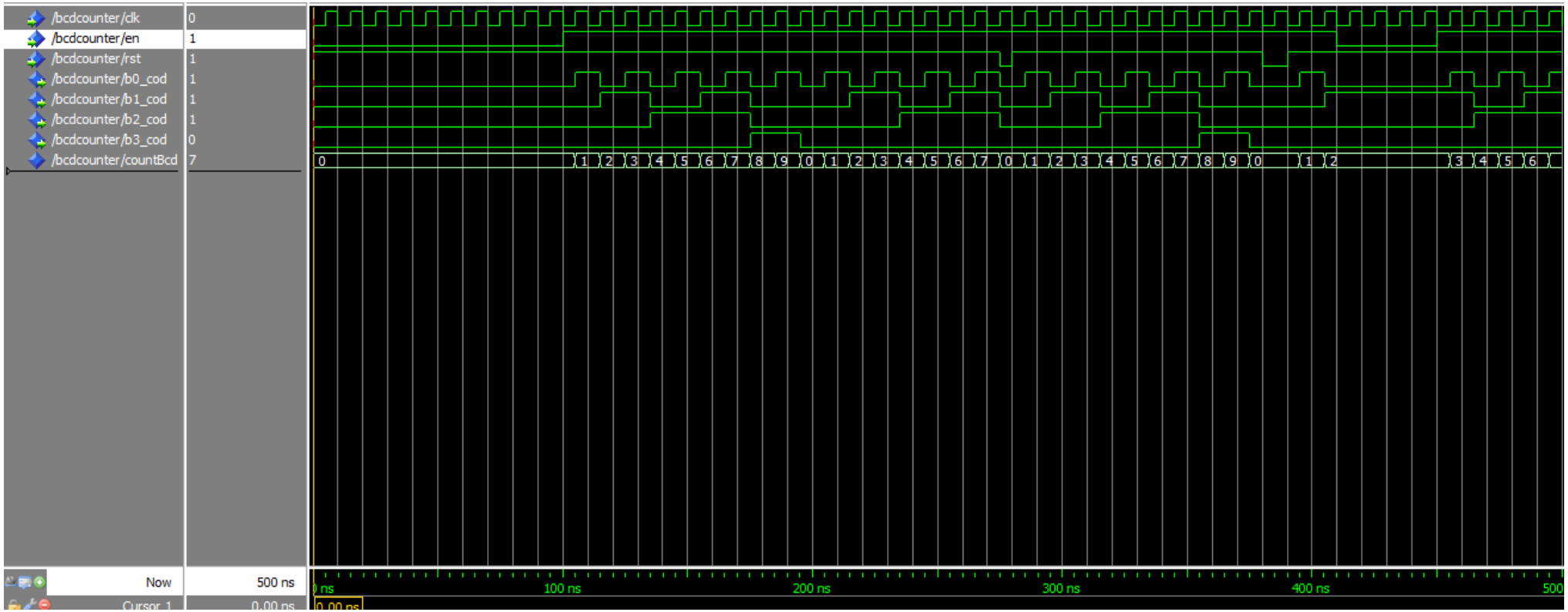

Abbildung 4 Coder test mit Modelsim

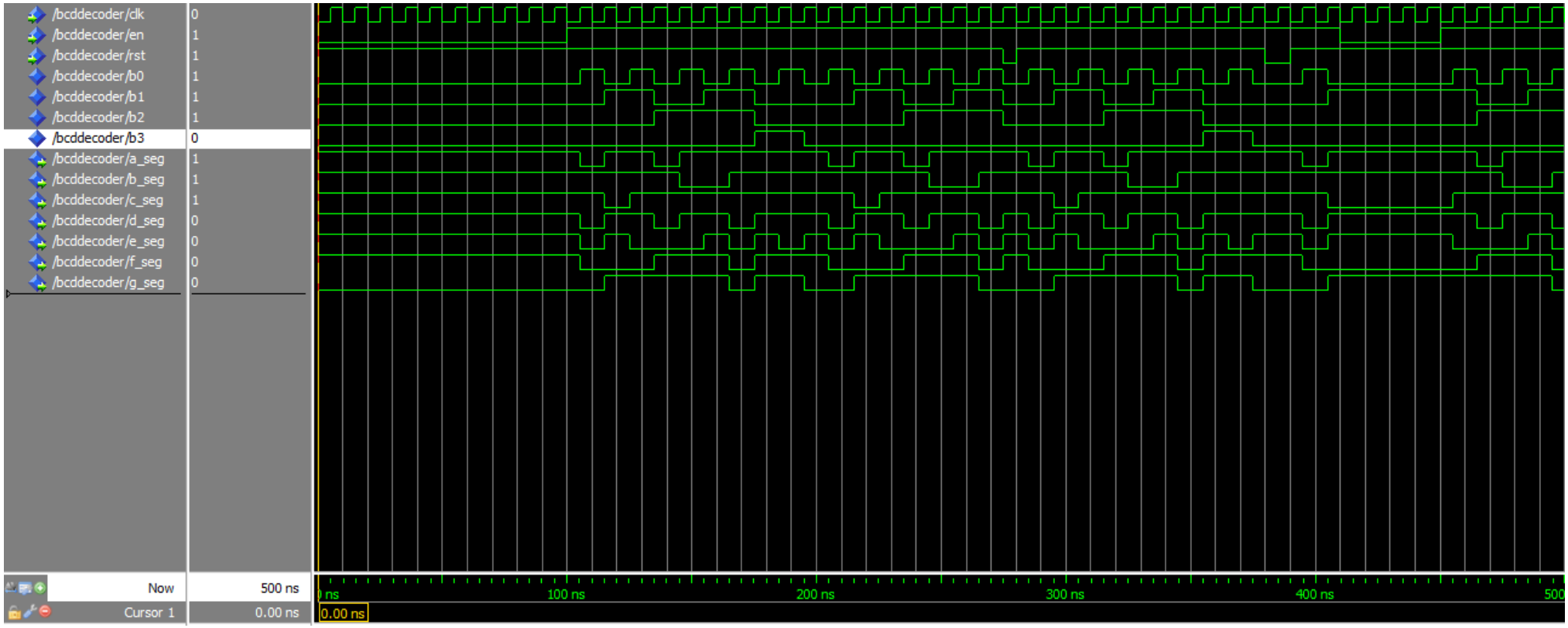

Abbildung 5 Decoder test

3.2 Simulation nach Synthese

In diesem Fall, habe ich die Projekte mit Quartus synthetisiert und dann die Simulation mit Modelsim gemacht. Diese Simulation ist nach Tools – Run Simulation Tool – Gate Level Simulation gestartet.

Abbildung 6 ist das Ergebnis dieser Simulation gezeichnet.

Man sieht, dass es mehr Signale gibt.

Abbildung 7 ist der Interpretation des Kode mit den Gattern (in Quartus: Tools-Netlist Viewer – RTL Viewer). Die Signale in Abbildung 6 sind die Signale von den Gattern, die man benutzen kann, wenn etwas nicht funktioniert.

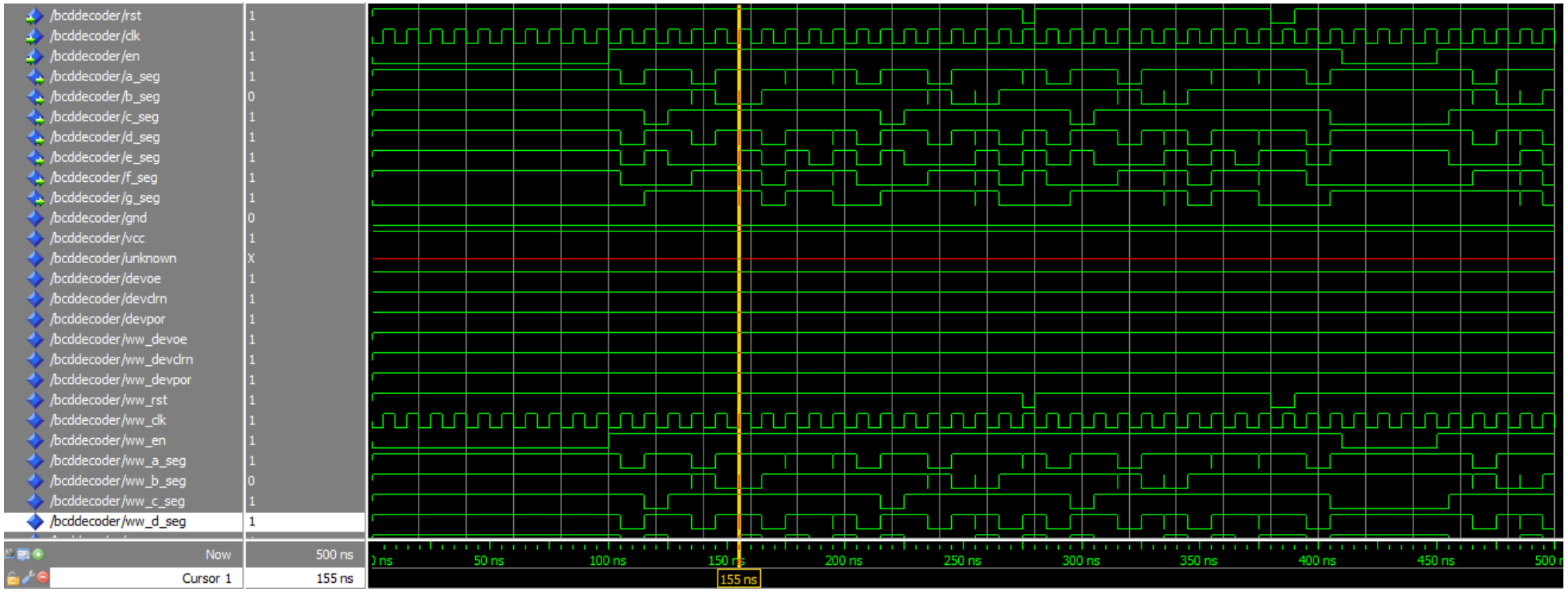

Abbildung 6 Decoder nach Synthese

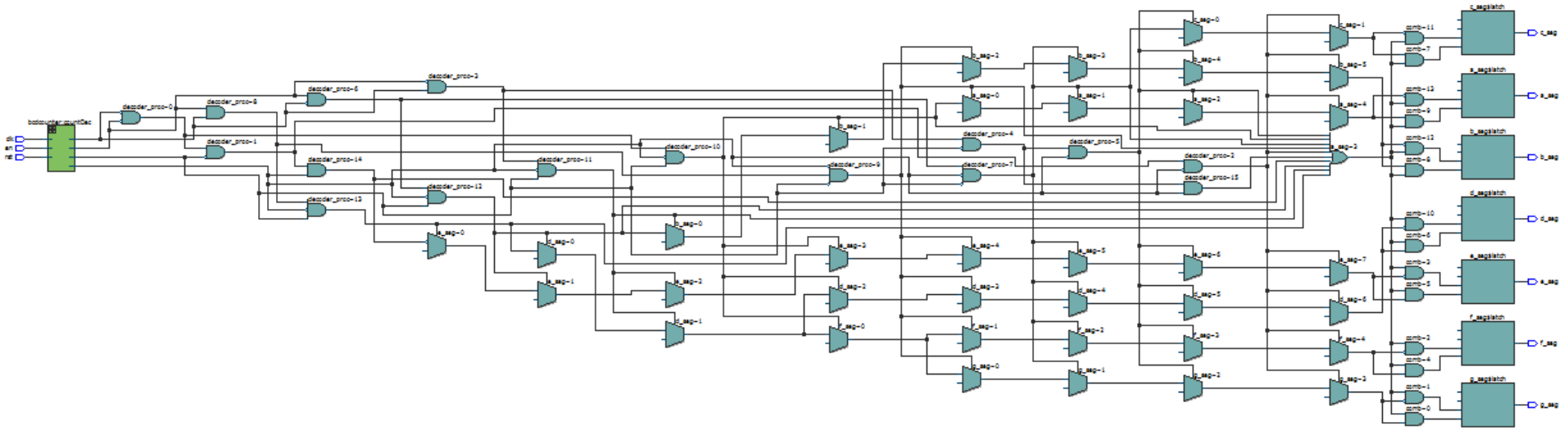

Abbildung 7 Interpretation der Kode mit den Gattern

4. Konklusion

Abschließend, habe ich von diesem Projekt viel über die VHDL gelernt. Ich habe auch Quartus geübt, um diese VHDL-Aufgabe zu entwerfen und auch ModelSim geübt, um meine Kode zuerst zu simulieren, vor ich diese auf dem Board lade.

Über die Aufgabe, sind die Spezifikationen einzuhalten, auch mit dem DE2-115-Development-Board.