Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering
InES Institute of
Embedded Systems

PA15_gelk_1 Polyphonic DDS Synthesizer mit MIDI Steuerung

Zürcher Hochschule für Angewandte Wissenschaften

Institute of Embedded Systems

Autoren	Katrin Bächli
Hauptbetreuer	
Nebenbetreuer	
Datum	12. Oktober 2015

Kontakt Adresse
c/o Inst. of Embedded Systems (InES)
Zürcher Hochschule für Angewandte Wissenschaften
Technikumstrasse 22
CH-8401 Winterthur

Tel.: +41 (0)58 934 75 25
Fax.: +41 (0)58 935 75 25

E-Mail: katrin.baechli@zhaw.ch
Homepage: `http://www.ines.zhaw.ch`

Inhaltsverzeichnis

1. Einleitung **3**

2. Glitches **4**
2.1. Definition Glitches 4
2.2. Ursache für Glitches 4
2.3. Glitches erzeugen 5
2.3.1. Glitches Aufgrund von Bauteiltoleranzen 5
2.3.2. Glitches Aufgrund von Pfadverzögerung 6

3. Metastabilität **8**
3.1. Problemanalyse 8
3.1.1. Ansatz 9
3.1.2. Implementation 9
3.2. bla 10
3.2.1. blu 10

4. Problembehandlung **11**
4.1. dringd 11

5. Implementation des Synthesizers **12**
5.0.1. blu 12

6. Resultate **13**
6.0.2. blu 13

A. Anhang 1: Englische Definitionen Glitches **I**

B. Anhang 2: VHDL-Code Glitch detect **II**
B.1. Bliblibli II

Liste der noch zu erledigenden Punkte

Totalüberarbeitung Text Bild: Deodieren 3,7,11,15. Kein RESET.(Blick auf Endlösung) 5
Bild Asynchronem Zähler besser beschreiben 6
Timeanalyse für FF 6
Bild FF1 verzögert, FF2 schneller 6
korrektes Wort ?(Concourent Assignment) 6

1. Einleitung

Die Projektarbeitet bietet die Möglichkeit, sich vertieft in VHDL einzuarbeiten. Mit vertieft ist das Erstellen eines eignen Projektes gemeint, wie auch das Kennenlernen der Spezialitäten dieser Sprache gemeint. Der erste Teil der Arbeit kümmerte sich um die Auseinandersetzung mit der Sprache VHDL und deren eigenen Herausforderungen. Glitches und Metastabilität werden herbeigeführt, damit man ihr Auftreten verstanden wird. Im zweiten Teil wird das Projekt, aus der Vorlesung Digitale Technik II, weiterentwickelt.

Ziel: VHDL Programmierung vertiefen mit zwei **Grundübungen** zu Glitches und Metastabilität. Danach ein Synthesizer mit FPGA realsieren.
Motivation: Tiefer Einblick in VHDL-Coderiung und deren Fehler.

Vorschlag Vorgehen:
1. Glitches reporduzieren
2. Metastaibiltät bei Detektion nachweisen

2. Glitches

2.1. Definition Glitches

Im technischem Bereich bedeutet ein Glitch, eine ungewollte, flüchtige Signalspitze, die ein Fehlverhalten im System verursacht (vlg.Cambridge Dictionaire in: A Definitionen Glitch in english). Am intuitivsten ist die bildliche Darstellung des Fehlverhaltens (siehe Abbildung 2.1).

Abbildung 2.1.: Glitch-Signalspitzen

Das Glitch ist in der digitalen Signalverarbeitung ein bekannter Begriff und wird dort unter anderem leicht sarkastisch beschrieben:
Als "Glitch"wird eine ungewollte, flüchtige SSignalspitze"bezeichnet, die Zähler aufwärts zählt, Register löscht oder einen ungewollten Prozess startet."(Fletcher, Digital design, 472).

Aus diesem Grund ist es wichtig, diesen Begriff und seine Ursache zu kennen.

2.2. Ursache für Glitches

Die Urache für der flüchtigen Spannungsspitzen sind asynchrone Inputs. Alle Prozesse, die nicht getaktet werden (z.B. In- und Output-Logik, direkte Signalzuweisungen) können ungewollte Spannungsspitzen verursachen.In Abbilung 2.2 verursacht das ungetaktete Eingangssingal Q die Spannungsspitzen, weil das Singal Q zu lange anhält.

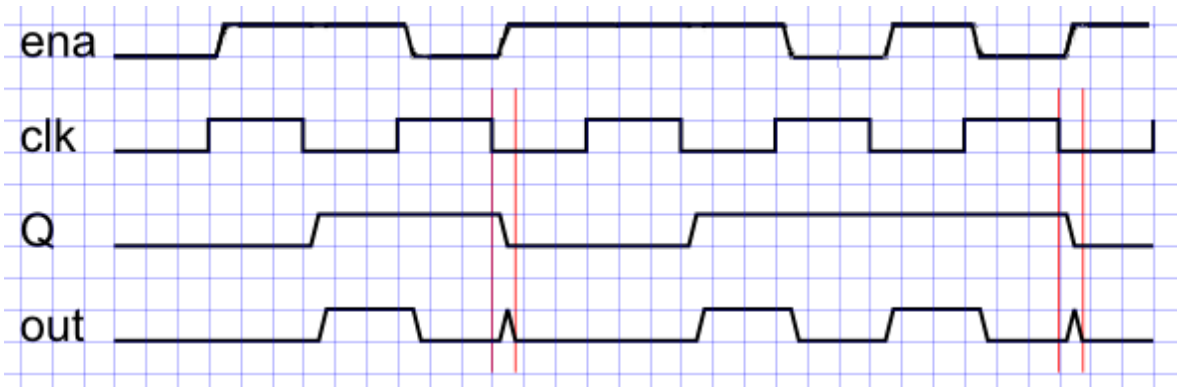

Abbildung 2.2.: Ursache für Glitches

2.3. Glitches erzeugen

Die erste Aufgabe der Projektarbeit ist, Glitches zu detektieren, um das Phänomen besser verstehen zu können.

2.3.1. Glitches Aufgrund von Bauteiltoleranzen

Der erste Ansatz war, ein Zähler aus 4 Flipflops mit asynchronem Dekoder zu implementieren. Die Erwartung war, dass durch die Bauteiltoleranzen der Flip-Flops die vier Ausgänge an den FlipFlops nicht genau der zu erwartenden nächsten Zahl entspricht, sondern kurzzeitig eine falsche Zahl am Dekoder anliegt. Dadurch zählt der Dekoder falsch.

Konzept

Um die Chance eines falschen Wertes zu erhöhen, werden vier Werte dekodiert: dezimal 3,7,11 und 15. Diese vier Werte folgen in denselben Abständen von 80 ns. Erscheint an den Ausgängen ungewollte eine dieser 4 Zahlen, so werden diese falscherweise dekodiert. Dies falschen Peaks sollten *zwischen* den regelmässigen Abständen auftreten.

Totalüberarb
Text Bild:
Deodieren
3,7,11,15. Ke
RESET.(Blic
auf Endlösur

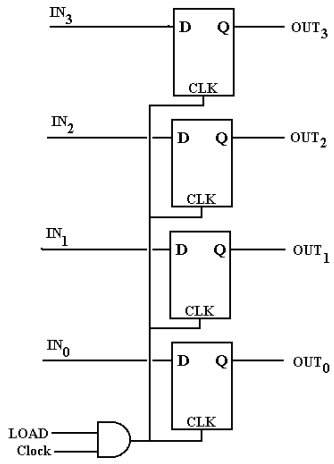

Abbildung 2.3.: RTL Zahler mit asynchronem Dekoder

Implementation

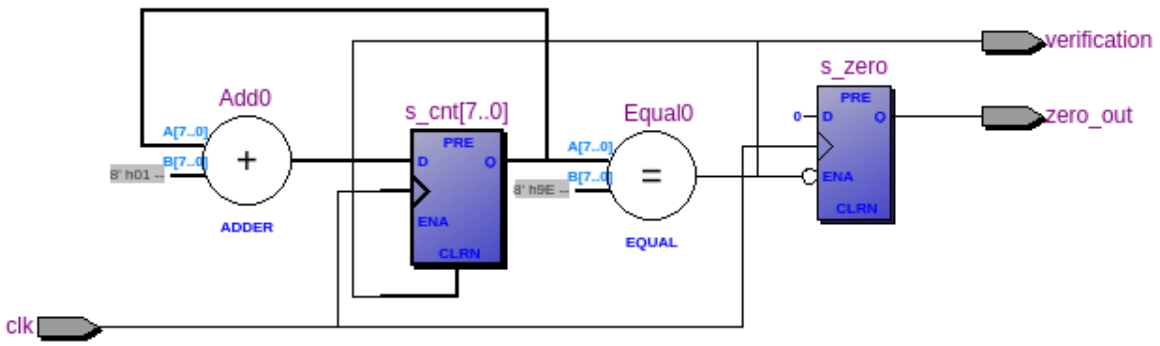

Abbildung 2.4.: RTL Zahler mit asynchronem Dekoder

Bild Asynch nem Zähler besser be- schreiben

Um die gewollten Zahlenwerte von den Glitches zu unterschieden, wird das asynchrone Singnal getaktet. Dadurch erscheint der korrekte Zählwert mit einem Takt Verzögerung. Die Periode ist 20 ns (CLK = 50 MHz).

Result

Der Ansatz, dass die Bauteiltoleranzen der Flip-flops eine Ursache für asynchrone Inputs in den Dekoder sind ist korrekt. Die Umsetzung zeigte sich jedoch als schwierig, da die heutigen Flip-Flops zu schnell sind bzw. ihre Toleranzen zu klein um sichtbar zu werden. Aus diesem Grund entschlüsselte der asynchrone Dekoder trotz kleinen Verzögerungen die Werte stets korrekt.

Timeanalyse für FF

2.3.2. Glitches Aufgrund von Pfadverzögerung

Der zweite Ansatz ist die gesuchte Bauteilverzögerungen über längere Signalpfade zu simulieren.Der Dekoder des Zählers bleibt asynchron.

Konzept

Dekodiert wird die Zahl 15. Durch intelligentes Routing (FF 1 wird verzögert, FF 2 wird beschleunigt) wird der Zustand der Zahl 11 forcier

Bild FF1 verzögert, FF schneller

Implementation

Cyclone II, Board De2. Quartus 13.0sp.

Die Pfad*verlängerung* wird über das Routing über die GPIO-Pins des Headers 1 gemacht (siehe Abbildung 2.6. Die obersten vier Doppel-Pins erhalten eine "Brücke", sodass das Signal links ausgegeben und rechts wieder eingespiesen wird.
Signal*verkürzung* ist eine direkte Signalzuweisung .

korrektes Wo ?(Concouren Assignment)

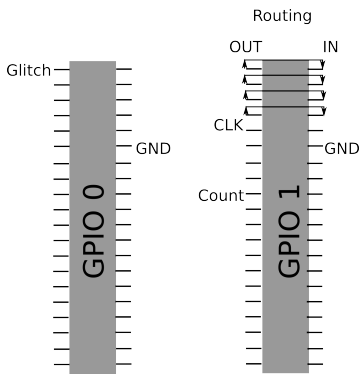

Abbildung 2.5.: GPIO Anschlüsse

Auf dem KO wird das asynchrone Glitch-Signal und das synchrone Zählersignal neben dem Takt ausgegeben. Weil der Zähler synchronisiert wurde, ist der Wert 1 Periode (= 20 ns) später als der Glitch.

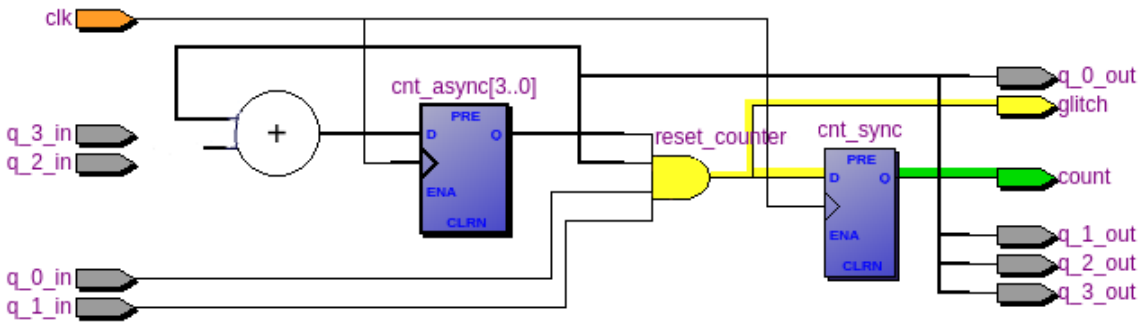

Abbildung 2.6.: Zähler mit Signal-Routing über GPIO

Im RTL-Diagramm sieht man deutlich den Unterschied zwischen dem asynchronen Zähler, der über das Gate *reset_counter* beim Wert 15 einen Impuls an den Ausgang glitch gibt und dem synchronisierten Zähler *cnt_sync* der dem asynchronen Ausgang nachgeschaltet ist und dieses Signal taktet. Das getaktete Zähl-Signal geht an den Ausgang *count*.

Result

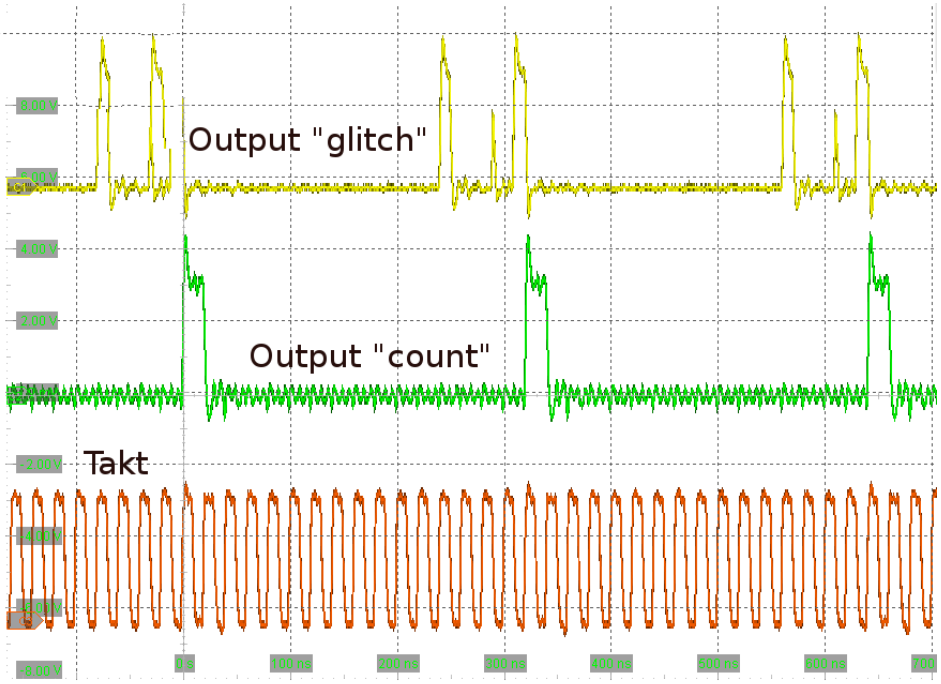

Abbildung 2.7.: Glitch (gelb), Zähler (grün) und Takt (orange)

Typisch ist, dass der synchrone Zähler eine Signalbreite von genau einer Periode hat, da dieses Signal getaktet ist. Dagegen hat der asynchrone Glitch keine konstante Breite.

1. Bei welchen Zählständen treten Glitches auf?
2. Wie hängen die Zählständen mit dem gewählten Routing zusammen?

3. Metastabilität

3.1. Problemanalyse

B.1 ??

Abbildung 3.1.: Bildbeschreibung ....

3.1.1. Ansatz

3.1.2. Implementation

Verbinden Sie sich als nächstes per Android Debug Bridge (ADB) mit der STB mit dem Kommando:

chapterBausteine eines Synthesizers

3.2. bla

B.1 **??**

Abbildung 3.2.: Bildbeschreibung ....

3.2.1. blu

Verbinden Sie sich als nächstes per Android Debug Bridge (ADB) mit der STB mit dem Kommando:

4. Problembehandlung

4.1. dringd

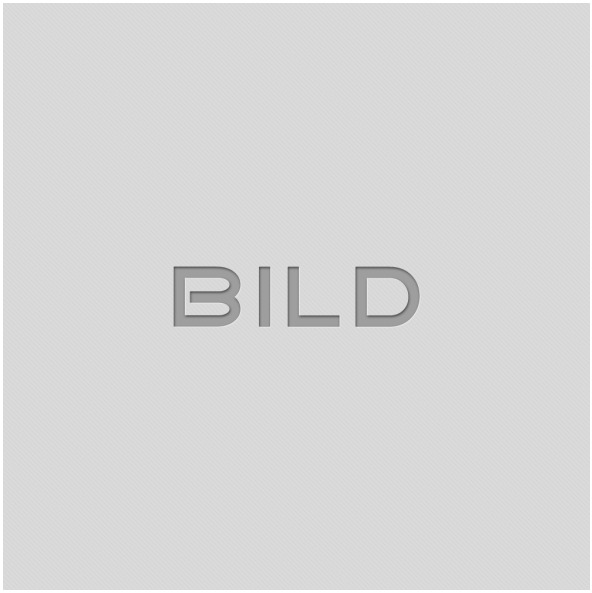

Abbildung 4.1.: blabla

5. Implementation des Synthesizers

B.1 **??**

Abbildung 5.1.: Bildbeschreibung ....

5.0.1. blu

Verbinden Sie sich als nächstes per Android Debug Bridge (ADB) mit der STB mit dem Kommando:

6. Resultate

B.1 **??**

Abbildung 6.1.: Bildbeschreibung ....

6.0.2. blu

Glossar

A. Anhang 1: Englische Definitionen Glitches

B. Anhang 2: VHDL-Code Glitch detect

B.1. Bliblibli

- sfasdfasdfasfasf