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	13. 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

Liste der noch zu erledigenden Punkte

1. Einleitung

Die Projektarbeitet bietet die Möglichkeit, sich vertieft in VHDL einzuarbeiten. Mit vertieft ist das Erstellen eines eignen Projektes wie auch das Kennenlernen der Eigenheiten der Sprache VHDL gemeint.

Der erste Teil der Projektarbeit umfasst die Auseinandersetzung mit der Sprache VHDL und deren eigenen Herausforderungen. Die zwei Fehlerquellen *Glitches* und *Metastabilität* werden künstlich erzeugt, damit ihre Ursache verstanden wird.

Der zweiten Teil der Projektarbeit beinhaltet eine Weiterentwicklung des Synthesizer-Projektes aud der Vorlesung Digitale Technik II. Konkret soll die Frequenzmodulation vertieft und der Midianschluss implementiert werden.

Eine Projektarbeit verdient nicht ihren Namen, wenn am Schluss der Arbeit nicht ausführlich die Resultate diskutiert und die verwendete Literatur genannt wird. Es folgt deshalb ein ausführlicher Anhang, in dem unter anderem auch der verwendete VHDL-Code abgebildet ist.

2. Glitches

2.1. Definition Glitches

Im technischem Bereich bedeutet gemäss Cambridge Dictionaire ein *glitch*, eine ungewollte, flüchtige Signalspitze, die ein Fehlverhalten im System verursacht. Im Anhang A befindet sich der Originaltext wie auch noch eine weitere Defintion aus dem englischen Sprachraum.

In der digitalen Signalverarbeitung ist das Glitch ein bekannter Begriff und wird dort unter anderem leicht sarkastisch beschrieben:

"Als "Glitch" wird eine ungewollte, flüchtige "Signalspitze" bezeichnet, die Zähler aufwärts zählt, Register löscht oder einen ungewollten Prozess startet." (Fletcher, Digital design, 472)

Am intuitivsten ist die bildliche Darstellung des soeben beschriebenen Fehlverhaltens (Abbildung 2.1). In dieser Signalabfolge treten zwei mal Glitches auf, die eigentlich nicht dort hingehören.

besseres Bild def Glitch

Abbildung 2.1.: Glitch-Signalspitzen

Auf den ersten Blick scheinen solch temporäre Spannungsspitzen nicht zu stören. Doch wenn man Pech hat, sind Glitches der Auslöser für Abstürze oder zumindest für ein Fehlverhalten eines Gerätes. Aus diesem Grund, wird nun der Ursache dieser Spitzen nachgegangen.

2.2. Ursache für Glitches

Der Auslöser der flüchtigen Spannungsspitzen sind asynchrone Inputs vor einem asynchronen Bauteil oder verzögerte Signale. Trifft z. B. vor einem Dekoder von vier Leitungen, das Signal einer Leitung zu spät an, entschlüsselt der Dekoder kurzfristig einen falschen Wert. Obwohl die Störung nur kurz ist, übermittelt ein asynchroner den falschen Wert direkt an seinen Ausgang.

2.2.1. Asynchroner Input

Das ungleichzeitige Eintreffen von Signalen kann z.B. durch lange Signalpfade (Leitungen), unterschiedliche Durchlaufverzögerungen der vorangehenden Flip-Flops oder unterschiedliche Logik-Zeiten entstehen. Grundsätzlich gelten alle nicht-getakteten Prozesse als potenzielles Risiko für Glitches, da man bei ungetakteten Prozessen nicht weiss, wie lange sie dauern.

2.2.2. Nachteil getakteter Prozesse

Jeder getaktete Prozess verzögert die Verarbeitung. Aus diesem Grund wird abgewogen, wo Prozesse getaktet und wo sie asynchron getätig werden. In VHDL gibt es viele asynchrone Vorgänge (wie ungetaktete Prozesse oder Singalzuweisungen), deshalb ist es vorteilhaft, wenn das Risiko asynchroner Prozesse bekannt ist.

Abbilung 2.2 zeigt ein leicht verzögertes (getaktetes) enable-Signal zu einem anders verzögerten (getakteten) Flip-Flop-Eingangssignal Q. Der Ausgang des Flip-Flops weist kurzzeitig Glitches auf.

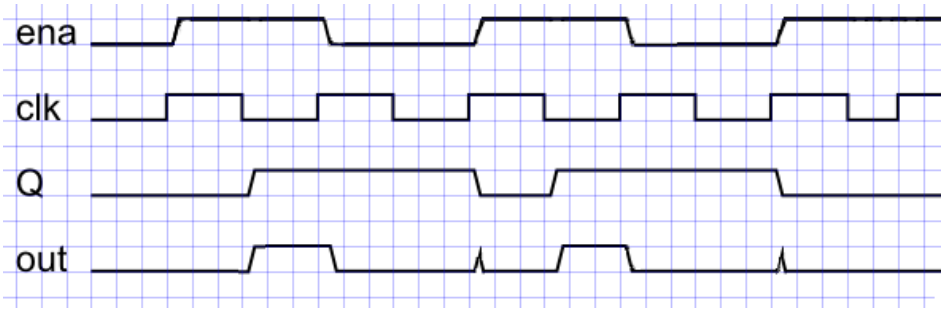

Abbildung 2.2.: Mögliches Bsp. für Glitches

2.3. Glitches erzeugen

Was in Glitch ist, ist relativ einfach zu beschreiben. Ein Glitch jedoch mit moderner Digitaltechnik zu erzeugen, erweist sich als etwas schwerer. Hier zwei Ansätze, die getestet werden.

2.3.1. Glitches Aufgrund von Bauteiltoleranzen

Der erste Ansatz ist, ein Zähler aus vier Flipflops mit asynchronem Dekoder zu implementieren.

Konzept

Die Erwartung ist, dass aufgrund der *Bauteiltoleranzen* der Flip-Flops die vier Ausgänge an den Flip-Flops nicht gleichzeitg ihren Wert übermitteln. Die einen sind leicht schneller, die anderen leicht verzögert. Dadurch ergibt sich kurzzeitig am asynchronen Dekoder einen falschen Wert.
Damit ein abnormaler Wert in einem Zähler erkannt wird, sendet der Dekoder bei der Zahl 7 einen Peak. Ohne Glitch entschlüsselt der Dekoder in regelmässigen Abständen von 160 ns diese Zahl. Aufgrund der Flip-Flop-Bauteiltolereanzen ist ein kurzzeitiges Dekodieren einer 7 *ausserhalb der Periode* T zu erwarten.

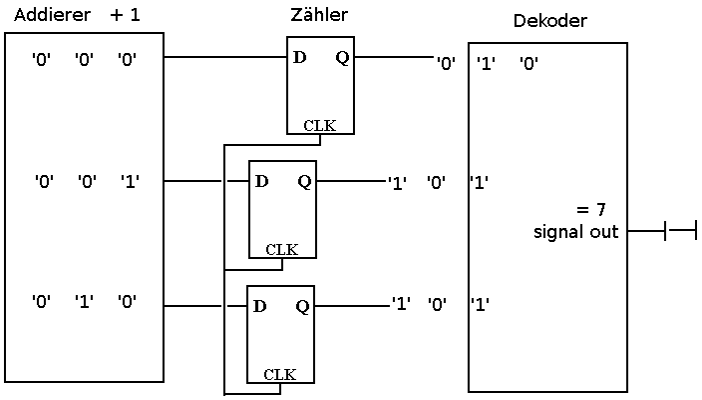

Abbildung 2.3.: Ausnutzen der Bauteilverzögerung

Implementation

Bild anpasse

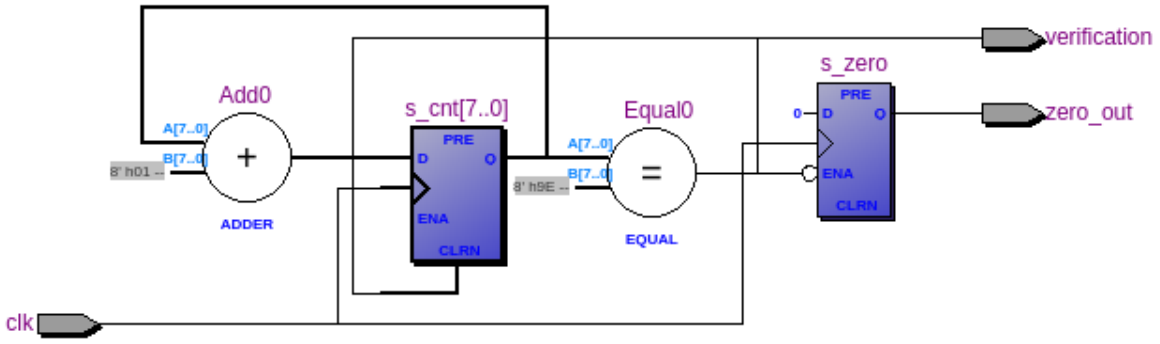

Abbildung 2.4.: RTL Zahler mit asynchronem Dekoder

Bild Asynchnem Zähler besser beschreiben

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)

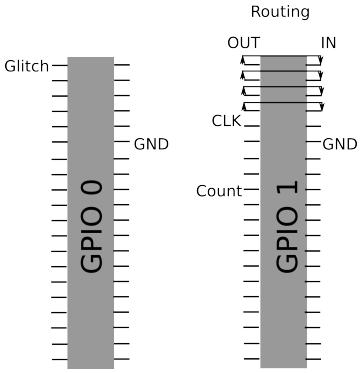

Abbildung 2.5.: GPIO Anschlüsse

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.

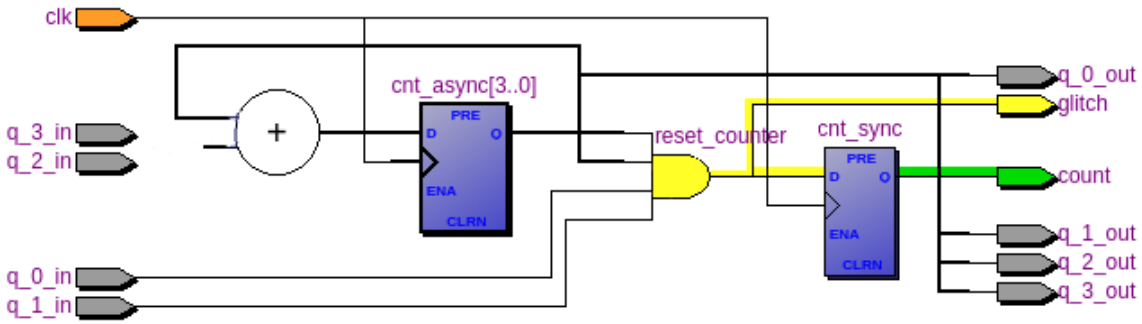

Abbildung 2.6.: Zähler mit Signal-Routing über GPIO

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

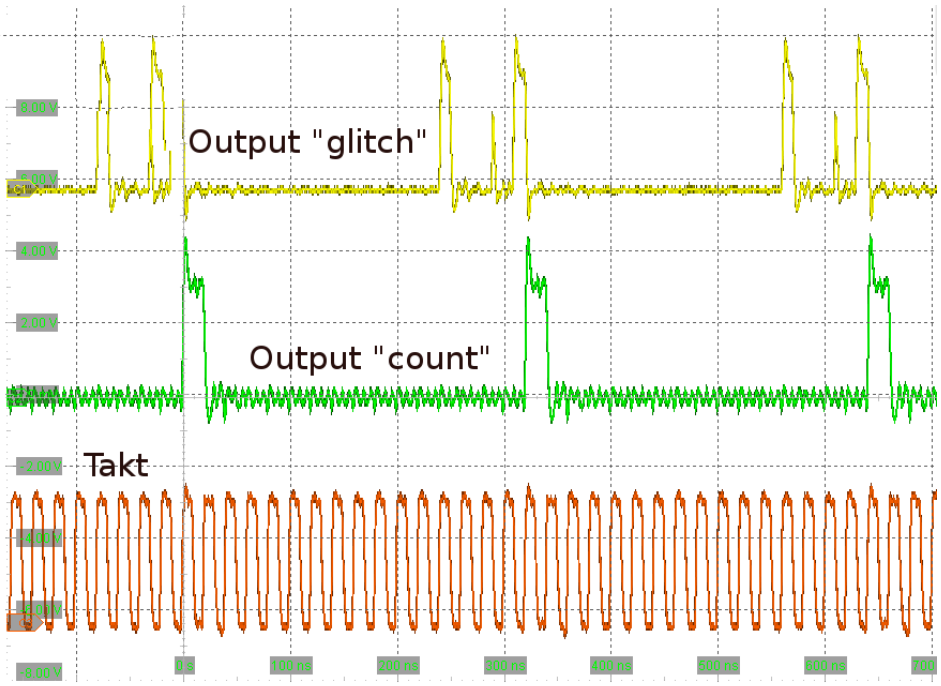

Abbildung 2.7.: Glitch (gelb), Zähler (grün) und Takt (orange)

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

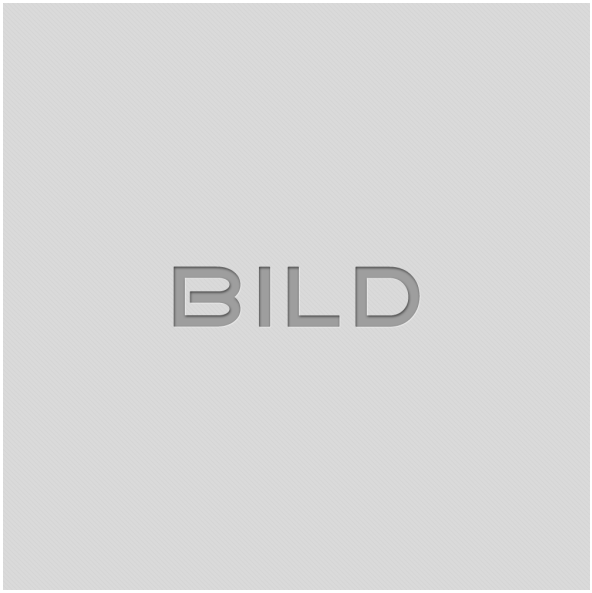

Abbildung 4.1.: blabla

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