DI - Hausarbeit

Fehlersichere Übertragung und Speicherung

Eric Antosch

2020

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel A

Plausibilitätsprüfung

A.1 Einleitung

Aufgabe 1.0

In dem ersten Teilbereich der Hausarbeit soll das von einem externen Sensor erfasste 10 Bit breite Datenpacket auf hinreichende Abtastung mittels einer Plausibilitätsprüfung überprüft werden. Dabei soll das Signal G_X überprüft und dann mit einem Signal G_{XOK} dargestellt werden, dass das Signal zur Weiterverarbeitung übertragen werden kann.

Wir wollen uns zuerst einmal anschauen, was der Gray-Code überhaupt ist, um eine Idee dafür zu bekommen, mit welchen Methoden die Aufgabe bewältigen können. Während bei dem in der Digitaltechnik sehr verbreitetem Binärcode die sogenannte Hamming-Distanz unterschiedlich groß sein kann, ist der Gray-Code so konzipiert, dass diese immer nur genau 1 beträgt. Die Hamming-Distanz beschreibt dabei die Distanz zwischen zwei aufeinanderfolgenden Zahlenwerten charakterisiert als die Differenz in den Ziffern der beiden Zahlen. Die Zahl $n = 3$ und zu der Zahl $n = 4$ haben die binären Darstellungen $n_B = 11_{(2)}$ und $n_B + 1 = 100_{(2)}$. Die Hamming-Distanz ist hier also 3, da sich drei Stellen der Zahl zu der nächsten ändern. Für die Messung von Messstrecken findet sich so keine einfache Methode, die Richtigkeit bzw. Plausibilität der Werte zu testen. Im Gegensatz dazu findet sich $n_G = 010_{(2G)}$ und $n_G + 1 = 110_{(2G)}$ mit einer Hamming-Distanz von 1. Hier können wir einfach überprüfen, ob der Wert, der als nächstes eingelesen wird, sich in der Hamming-Distanz um 1 von dem vorherigen verändert hat. Wenn nicht, so stimmt die Messung nicht vollständig oder die Auflösung ist nicht hoch genug. Besonders bei Messungen von Werten, die keine allzu starken Schwankungen erlauben, ist diese Art der Plausibilitätsprüfung sehr sinnvoll.

A.2 Blockschaltbild

Wir wollen nun zunächst das Blockschaltbild für unser Vorhaben erstellen, sodass wir bei Beschreibung des Systems in VHDL einen besseren Überblick über alle Signale und Komponenten haben.

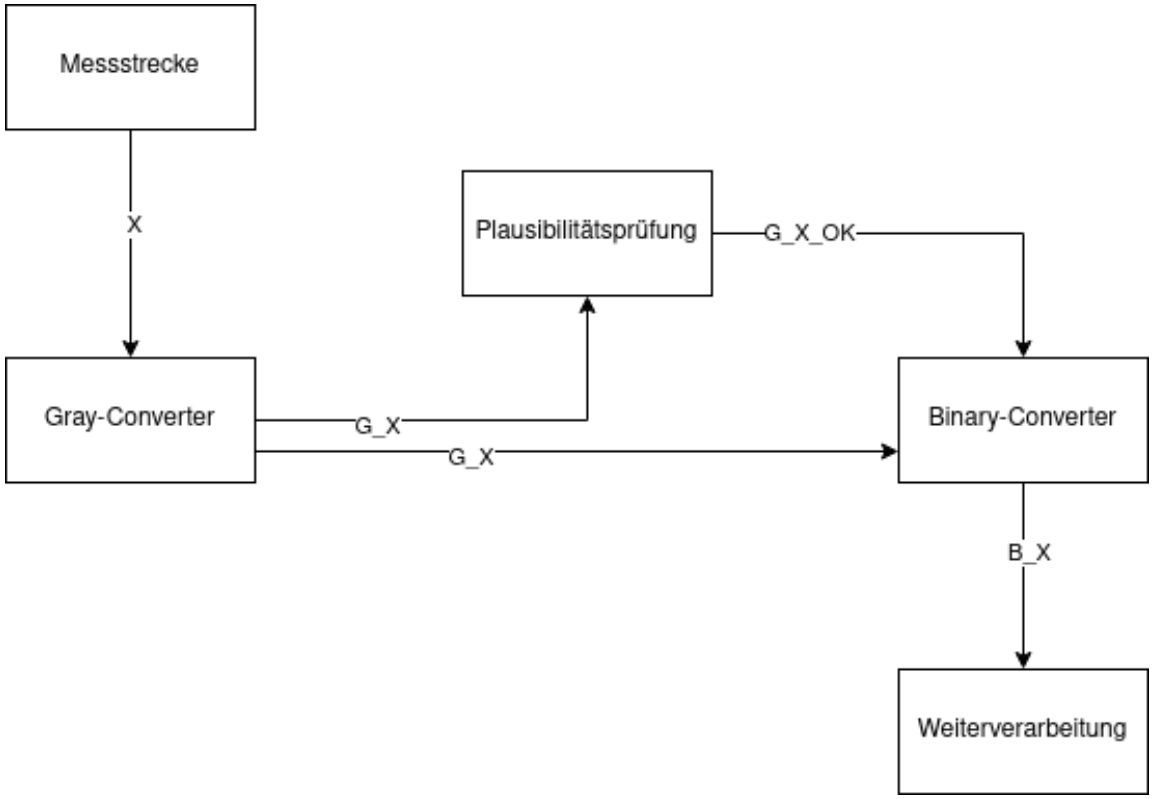

Abbildung A.1: Blockschaltbild der Aufgabe A zur Plausibilitätsprüfung

A.3 Lösungsidee

Wie schon im Einleitungstext erwähnt, bietet es sich bei dir Aufgabe an, mithilfe der Eigenschaften des Gray-Codes eine Überprüfung der Plausibilität durchzuführen. Dafür wollen nehmen wir unser Eingangssignal G_X und speichern dies zunächst auf ein Signal buf, um es später mit dem im nächsten Zyklus eingelesenem Eingangssignal zu vergleichen. Basierend auf dem Ergebnis der Hamming-Distanz-Berechnung setzen wir G_{XOK} also entweder auf True oder False. Das eingelesene Signal G_X wird das neue buf und der Prozess wiederholt sich.

A.4 Beschreibung in VHDL

A.5 Simulation der Ergebnisse

Da wir unsere Schaltung in ModelSim simulieren, verwenden wir eine .do-Datei, um den Verlauf der Schaltung zu simulieren. Wir wollen dabei sowohl den Fall erproben, dass die Daten richtig sind, als auch, dass die Daten zuweit voneinander entfernt sind, um plausibel zu sein.

A.6 Fazit

Wir können anhand unserer Ergebnisse einige Dinge feststellen. Zum einen ist es wirklich erstaunlich, wie die Eigenschaften von speziellen Codes die Verarbeitung und Implementierung von bestimmten Lösungen vereinfacht. Die Logik, um eine solche Plausibilitätsprüfung durchzuführen, wenn die Eingangsdaten zum Beispiel weiterhin als Binärcode kodiert sind, wäre um einiges komplizierter und außerdem nicht wirklich effizient oder wirtschaftlich. Es bietet sich also stark an, die Möglichkeit von den verschiedenen Codes aus der Digitaltechnik einzustudieren. Es ist außerdem ersichtlich, dass sich eine Darstellung von Problemstellungen nach dem gelernten Schema anbietet. Es lässt einen stark über die einzelnen Komponenten und die Zusammenarbeit auf einem tatsächlichen FPGA-Board nachdenken.

Kapitel B

Berechnung der Regelabweichung

B.1 Einleitung

Aufgabe 1.0

In der nächsten Aufgabe soll es um die Berechnung der Regelabweichung von zwei Signalen mithilfe eines Carry-Lookahead-Addierers gehen. Dabei ist unser auf Plausibilität geprüfte Signal B_X mit einem Sollwert B_W zu vergleichen und die Differenz auf einer bidirektionalen 10-bit breiten Busleitung auszugeben. Ebenfalls sollen die dedizierten Signale EXAKT und ZU_KLEIN seperat je nach Ergebnis ausgegeben werden.

B.2 Blockschaltbild

B.3 Automatengraph

B.4 Lösungsidee

Wir wollen uns nun einmal anschauen, mit welchen Mitteln wir unser Ziel erreichen können.

B.4.1 Carry-Lookahead-Addierer

Bei dem Carry-Lookahead-Addierer, den wir für unsere Berechnung der Regelabweichung verwenden wollen, handelt sich um eine Weiterentwicklung des Carry-Ripple-Addierers. Hier werden also nicht einfach nur mehrere Volladdierer miteinander so verschaltet, dass mehrere Bit breite Signale miteinander addiert werden können, sondern es gibt ein komplizierteres System, um die Übertrage der Volladdierer zu bestimmen. Fundamental für diese Überlegung sind die Konzepte von Generate, Propagate und Absorb.

Generate Bei Generate handelt sich um ein logisches Prädikat, welches wir im Folgenden durch $G(A, B)$ bezeichnen möchten mit $G(A, B) = A \cdot B$, wobei $\cdot$ das logische Und darstellt. Wir sagen also, dass zwei Ziffern genau dann generieren, wenn die beiden Ziffern immer einen Übertrag bilden und dieser nicht abhängig ist, ob ein weiterer Übertrag auf die Ziffern hinzuaddiert werden.

Propagate Bei Propagate handelt es sich ebenfalls um ein logisches Prädikat, welches wir im Folgenden mit $P(A, B) = A + B$ bezeichnen, wobei $+$ das logische Oder darstellt. Wir sagen also, dass zwei Ziffern genau dann propagieren, wenn die beiden Ziffern nur dann einen Übertrag bilden, wenn ein weiterer Übertrag von der Berechnung vorher kommt.

Absorb Wir wollen im folgenden Absorb als denjenigen Zustand definieren, der einen Übertrag von der vorherigen Berechnung absorbieren würde, dabei allerdings keinen Übertrag generiert. Mit $A(A, B) = \not(A + B)$ definieren wir einen Zustand, der das Gegenteil von Propagate darstellt und nur dann wahr ist, wenn beide Ziffern (im Falle einer binären Addition) 0 sind.

B.4.2 Zweier Komplement

Wenn wir die Differenz von zwei binären Zahlen berechnen wollen, müssen wir eine der beiden, da unser Carry-Lookahead-Addierer eigentlich nur Addition beherrscht, in ein anderes Format umwandeln, um eine Differenz berechnen zu können. Wir entscheiden uns hier für das Zweier Komplement, um aus einer der beiden Signalen, die eine binäre Zahl darstellen, eine negative Zahl zu machen, und diese dann zu addieren.

Bei der Umwandlung gehen wir dabei nach folgendem Schema vor:

1. Das Most-Significant-Bit stellt im zweier Komplement das Vorzeichen dar, wir setzen dies also auf eine 1, um ein Minus-Zeichen darzustellen
2. Wir invertieren alle weiteren Bits, alle 1 werden zu 0 und umgekehrt.
3. Zum Schluss addieren wir auf die Zahl eine 1, unsere Zahl ist damit umgewandelt.

B.4.3 Taktsynchronität

Wir müssen uns nun noch mit der Anforderung an unsere Lösung beschäftigen, taktsynchron und in Harmonie mit dem Busarbiter zu funktionieren. Dafür wollen wir in unserer Beschreibung mit VHDL eine eigene Komponente einrichten, die sich um genau diese Anforderungen kümmert.

B.5 Beschreibung in VHDL

B.6 Simulation der Ergebnisse

B.7 Fazit