ERA-Praktikum 2018

Digital Audio – Pegelanzeige

-Seriell/Parallel-Konverter Implementierung

Gruppennummer: 53

Betreuer: Beneikt Pfleiderer

Teilnehmer: Marc Sinner, Daniel Osipishin, Adrian Kögl

Inhaltsverzeichnis:

Aufgabenverteilung:

Projektleitung: Marc Sinner

Dokumentation: Adrian Kögl

Vortrag: Daniel Osipishin

Zeitplanung:

Abgabetermine:

Spezifikation: 13.05.2018

Implementierung: 17.06.2018

Ausarbeitung: 08.07.2018

Vortrag: 23.07 - 03.08.2018

(generell: Fertigstellung bis zum Mittwoch vor der Abgabe, um bei Rückfragen Betreuer bei der Sprechstunde zu fragen)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Marc Sinner | Zeit | Daniel Osipishin | Zeit | Adrian Kögl | Zeit |
| Planung und Erstellung der Protokolle | Bis 1.7.18 | Vorbereitung des Vortrags (Powerpoint, Vorzeigeversion, etc.) | 8.7.18 | Ausarbeitung der Dokumentation | Bis  1.7.18 |
| Erstellung des GIT-Repositories | Bis. 2.5.18 | Präsentation des Vortrags vorbereiten | 15.7.18 |  |  |
| Implementierung | Bis 10.6.18 | Implementierung | Bis 10.6.18 | Implementierung | Bis 10.6.18 |
|  |  |  |  |  |  |

Aufgabenbeschreibung:

Für eine digitale Audio-Pegelanzeige welche aus digitalen 18-Bit Audiodaten eine Pegelanzeige erstellt soll ein VHDL Programm für den Seriell/Parallel Konverter implementiert werden. Dieser Konverter ist die erster Komponente der Schaltung und bekommt als Eingang einen seriellen Datenstrom welchen es unter Berücksichtigung zweier Takte in zwei 18-Bit Werte umwandeln soll. Zudem soll nach jedem 18-Bit Paar ein Flag für die nächste Komponente in der Schaltung gesetzt werden.

Ist-Analyse:

- Erstellung einer Schaltungsskizze für den Seriell/Parallel Konverter

- zwei Lösungsansätze

- Prototyp der beiden Schaltungen in VHDL (kompiliert bereits aber funktioniert noch nicht)

Soll-Analyse:

-Implementation eines Lösungsansatzes für den Seriell/Parallel Konverter inklusive erfolgreicher Funktion anhand von Beispiel Datensätzen.

Erfolgskriterium:

Es soll der serielle Datenstrom (input) erfolgreich in zwei 18-Bit werten gespeichert werden mit Berücksichtigung der Takte.

Lösungsansätze:

Es gibt im Prinzip nur eine Möglichkeit die Implementierung zu variieren und zwar bei der Art der Speicherung der beiden 18 Bit werte.

Lösungsansatz Shift-Methode:

Hierbei wird ein 18 Bit großer Standart Logic Vektor verwendet. Und der neue zu hinzufügende Bit rechts an die den Vektor drangehängt, was letztendlich wie ein „Shift left“ funktioniert.

leftMemo <= leftMemo(17 DOWNTO 1) & sdata; --leftMemo: Vektor, sdata: Bit

Lösungsansatz Bit-Adressierung:

Hierbei wird ebenfalls ein 18 Bit großer Standart Logic Vektor erstellt. Jedoch wird beim Hinzufügen des aktuellen Bits die Stelle des Vektors adressiert und dort hinzugefügt. Dies wird mithilfe des counters gemacht, welcher angibt, das wievielte Bit momentan verarbeitet wird.

leftMemo(counter) <= sdata;

Verwendeter Lösungsansatz:

//todo