Entwicklung eines FPGA-basierten Prozessors

Die Idee

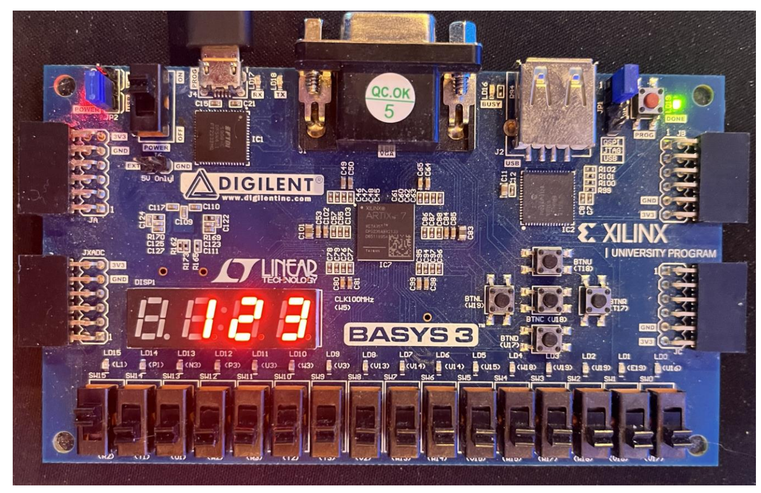
Wir leben in einer Zeit, in der Computer unglaubliche Rechenleistungen erbringen – doch die meisten Menschen wissen kaum noch, wie sie eigentlich funktionieren. Selbst wer programmieren kann, versteht oft nicht, was hinter den Kulissen passiert, weil moderne Programmiersprachen stark abstrahiert sind. Man schreibt Code nicht direkt für den Computer, sondern nutzt mehrere Schichten von Übersetzungen.

Um wirklich zu verstehen, wie ein Computer auf der **untersten Ebene** arbeitet, habe ich beschlossen, einen eigenen Prozessor zu entwerfen und ihn mithilfe eines sogenannten **FPGA** in der Praxis zu testen. Während dieses Projekts wurde mir klar, dass es nicht nur mir hilft, die Grundlagen besser zu verstehen, sondern auch für Schüler:innen eine spannende Möglichkeit sein könnte, in die Welt der Computer einzutauchen.

Deshalb lege ich besonderen Wert auf eine klare Dokumentation, einfache Bedienung und gute **Debugging-Werkzeuge**, welche es erlauben, die internen Vorgänge des Prozessor zu verlangsamen und in Echtzeit zu verfolgen. In Zukunft möchte ich die Benutzerfreundlichkeit noch weiter verbessern und mein Projekt als praktischen Einstieg in die Computerarchitektur für den Informatikunterricht nutzbar machen.

Was ist ein FPGA?

Ein FPGA (Field-Programmable Gate Array) ist ein spezieller Computerchip, der nachträglich programmiert werden kann, um verschiedene Schaltkreise oder Funktionen auszuführen. Anders als normale Prozessoren, die festgelegte Befehle abarbeiten, kann ein FPGA individuell konfiguriert werden, um genau die gewünschte Hardware-Schaltung darzustellen. Dadurch sind sie extrem flexibel und werden oft in Bereichen wie Signalverarbeitung, KI-Beschleunigung oder Prototypenentwicklung eingesetzt. Man kann sie sich wie einen Baukasten für digitale Schaltungen vorstellen, den man immer wieder neu zusammensetzen kann. Ich benutze ein FPGA in meinem Projekt, um die von mir entwickelte Hardwarearchitektur in der Praxis zu testen zu können.



FPGA Entwicklungs-Board

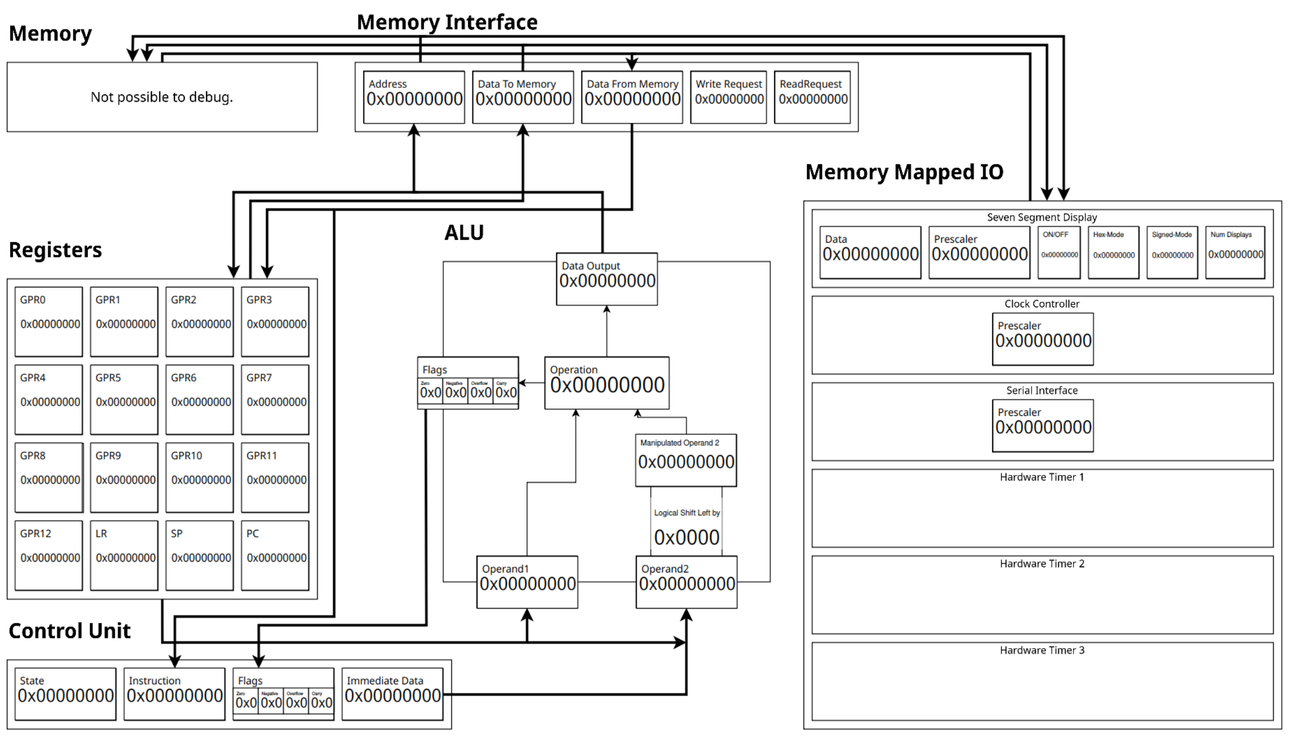
Programmieren ohne Abstraktion

Moderne Programmiersprachen nutzen mehrere Übersetzungsschichten, damit Menschen einfacher mit Computern arbeiten können. Das bedeutet, dass der Code, den man schreibt, nicht direkt von der Hardware verstanden wird. Stattdessen wird er zunächst in abstraktere Befehle umgewandelt, bevor er schließlich als Nullen und Einsen (Maschinencode) vom Prozessor ausgeführt wird. Diese Abstraktion erleichtert das Programmieren erheblich – man muss nicht genau verstehen, wie der Computer intern arbeitet. Dennoch kann ein tieferes Verständnis helfen, effizienteren Code zu schreiben und komplexe Probleme besser zu lösen. Deshalb lohnt es sich, für Lernzwecke ganz unten anzufangen und direkt mit dem Prozessor zu "sprechen".
Ein Beispiel dafür ist Assembly, eine Sprache, in der jede Anweisung direkt in eine binäre Repräsentation übersetzt wird, die der Prozessor versteht.

```
define displayControlAddress 0x40000050
define displayDataAddress    0x40000054

define displayControl        0b00000000111101000010010000100011
define displayData           123

start:

    ;write control data to 7-Segment Display
    MOV R0, displayControlAddress
    MOV R7, displayControl
    STOREW R7, [R0]

    ;write display data to 7-Segment Display
    MOV R0, displayDataAddress
    MOV R7, displayData
    STOREW R7, [R0]
```

Assembly Code um die Zahl 123 auf dem 7-Segment Display (siehe Bild oben) anzuzeigen.

Echtzeitüberwachung des Prozessors



Link zu meinem GitHub Repository