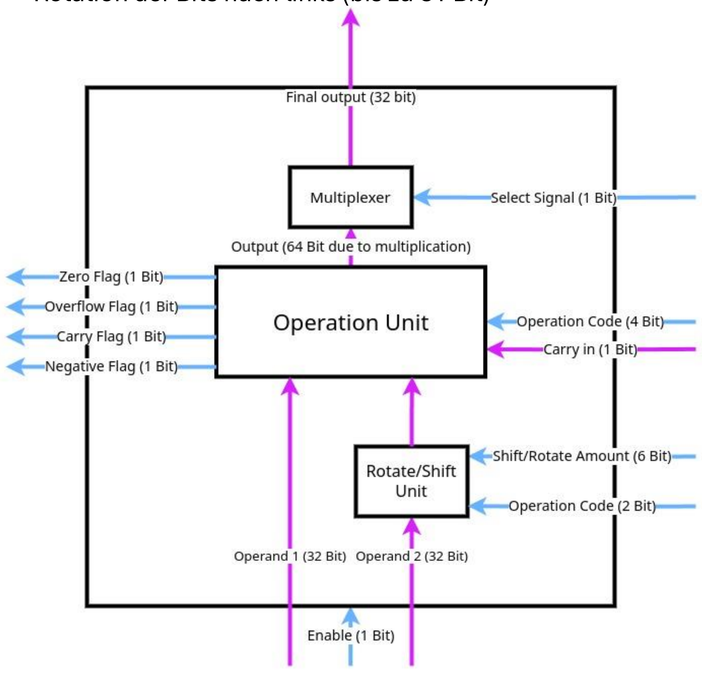
Die Technischen Details

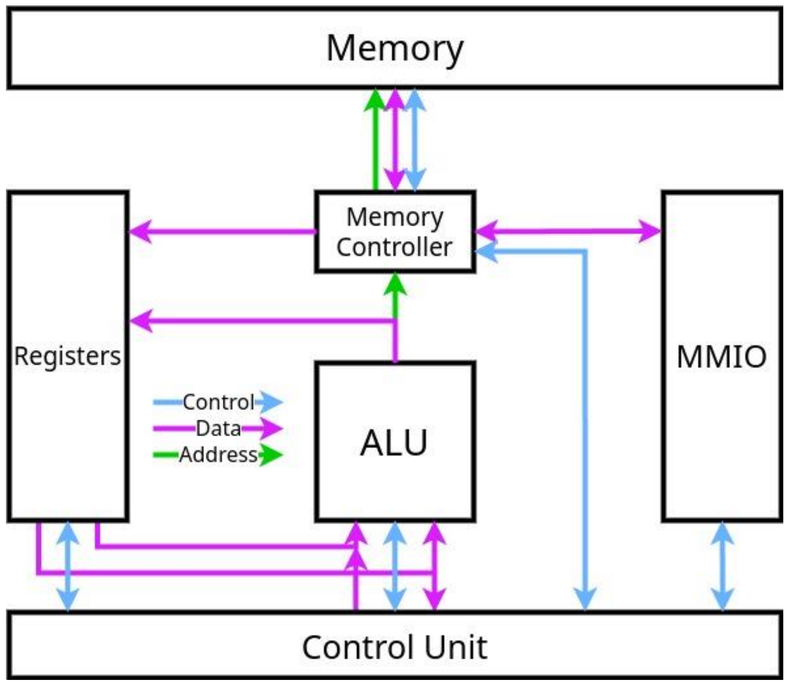
Arithmetic Logic Unit (ALU)

- **Unterstützung von folgenden arithmetischen Fixed-Point Operationen:**
  - Addition (mit und ohne Carry)
  - Subtraktion (mit und ohne Carry, regulär und reverse)
  - Multiplikation (64 Bit, wobei entweder die oberen oder die unteren 32 Bit ausgegeben werden können)
- **Unterstützung von folgenden logischen Operationen:**
  - AND, XOR, OR, NOT, AND NOT
- **Rotate/Shift Unit, die folgende Bit-Manipulationen an Operand 2 vor der Berechnung erlaubt:**
  - Logische Verschiebung nach rechts/links (bis zu 31 Bit)
  - Arithmetische Verschiebung nach rechts (bis zu 31 Bit)
  - Rotation der Bits nach links (bis zu 31 Bit)



Hardware-Überblick

- **32-Bit Von-Neumann-Architektur** entwickelt mit der Hardwarebeschreibungssprache **VHDL**
- Prozessor Kern mit **16 Registern**, **Kontrolleinheit** und **ALU**
- **RISC-basierter Befehlssatz** mit ungefähr 40 Instruktionen
- Maximale Taktrate von 50MHz mit der Option, diese beliebig nach unten anzupassen.
- Möglichkeit, Taktflanken per Knopfdruck zu triggern (**Debugging**).
- Emulation der Architektur auf einem **Basys 3 Board** von **Digilent**
- Menge des Hauptspeichers aufgrund der Hardware auf 225 KB begrenzt



Memory Mapped IO (MMIO)

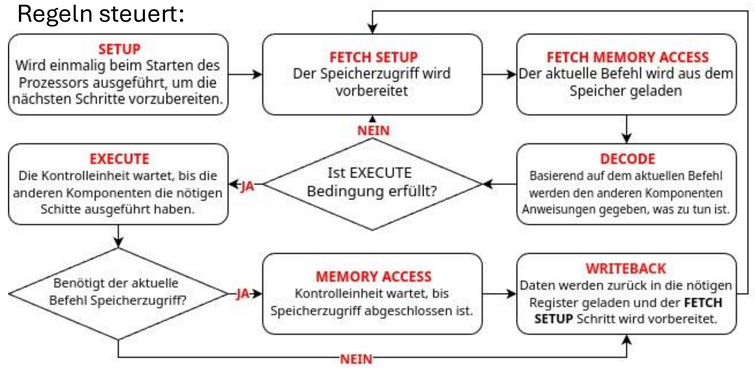
MMIO bietet eine gute Möglichkeit, externe Geräte wie Schnittstellen an den Prozessor-Kern anzubinden und auch in Zukunft Erweiterungen ohne großen Umbau der Architektur zuzulassen. Jedes MMIO-Gerät bekommt seine eigene Speicheradresse, welche nicht im Adressbereich des Hauptspeichers liegt. Der Prozessor-Kern kann dann durch einfachen Speicherzugriffe mit diesen Geräten kommunizieren. Dabei entscheidet in meinem Fall der Speichercontroller, ob ein Speicherzugriff an ein MMIO-Gerät oder den Hauptspeicher weitergeleitet werden soll.

Folgende MMIO-Geräte werden momentan unterstützt:

- Serielle Schnittstelle (UART)
- Mehrere Hardware-Timer
- 7-Segment-Display

Kontrolleinheit

Die Kontrolleinheit ist für die Verarbeitung von Befehlen und die Steuerung aller anderen Komponenten verantwortlich. Um dies umsetzen zu können, verfügt sie intern über einen Zustandsautomaten, der die internen Abläufe nach festgelegten Regeln steuert:



Programming

Programme für meinen Prozessor können in einer an ARM orientierten Assembly Sprache geschrieben werden. Sie bietet einen guten Einstieg in hardwarenahe Programmierung, da es eine überschaubare Anzahl an Befehlen gibt, die aber dennoch viel Potenzial für Optimierung bieten.

Assembling

Mit einem von mir geschriebenen Assembler, lässt sich der Programmcode in 32-Bit Instruktionen übersetzen, welche für den Prozessor verständlich sind. Über die serielle Schnittstelle können diese dann auf den Prozessor hochgeladen werden. Dabei werden die empfangenen Befehle durch den prozessorinternen Bootloader auf Fehler überprüft und anschließend in den Hauptspeicher geladen.

Debugging

Der Prozessor kann in einen Debug-Modus versetzt werden, in welchem er über die serielle Schnittstelle alle relevanten internen Signale an einen Computer sendet, welcher diese dann mithilfe von einer passenden Software visualisiert. Außerdem lässt sich die Taktrate des Prozessors beliebig verringern. Auch das manuelle Ausführen von Takten per Knopfdruck ist möglich, um Schritt für Schritt die Funktionsweise des laufenden Programmes überprüfen zu können.