Computer Design

Entwurf eines einfachen Ein-Takt-Prozessors in VHDL

Computer Engineering

Manuel Bergler (s0536440)
Hidayat Halim (s0536950)

Inhaltsverzeichnis

1. Einleitung

Das Ziel dieser Belegarbeit war das Erstellen eines funktionsfähigen Ein-Takt-Prozessors, der die Befehle des Xilinx picoBlaze (KCPSM3) ausführen kann.

Die Entwicklung erfolgte unter folgenden Annahmen:

- der Prozessor arbeitet pro Taktperiode einen Befehl ab
- die Länge der Taktperiode richtet sich nach der längsten möglichen Bearbeitungszeit eines Befehls
- nur die benötigten Teilschritte werden ausgeführt
- serielle Abarbeitung der Befehle
- 16 allgemeine Register vorhanden
- ein Reset-Eingang zur Initialisierung des Prozessors

Zusätzlich sollte eine Rahmenbedingung gesetzt werden, nach welcher der Prozessor optimiert werden soll.

Von uns wurden hierbei folgende Ziele gesetzt:

- Verwendung von wenig Fläche auf dem FPGA: , durch Verwendung von verteiltem Dual Port RAM im Flagregister und Verwendung eines Carry-Chain-Addierers in der Arithmetic Unit
- zeitlich effiziente Umsetzung: durch Verwendung des Carry-Chain-Addierers
- Modularität: der Prozessor wurde im Bottom-Up-Entwurf von uns programmiert und getestet, die oberste Prozessor-Einheit beinhaltet dabei alle Module in sich, jedoch sind diese alle in einzelne Dateien untergliedert und beinhalten ihre eigenen Testbenches

Die Ausführung von Interrupt- und Unterprogrammaufrufen ist in unserer Implementierung aus Zeitgründen nicht vollständig umgesetzt. Die Signale und Komponenten sind teilweise bereits vorhanden, wurden jedoch nicht mit dem Prozessor verbunden.

2. Befehlssatz

Im Folgenden wird kurz beschrieben welche Teilschritte für die einzelnen Befehlsgruppen im Datenpfad notwendig durchlaufen werden müssen.

Lade- und Speicherbefehle

Befehle	IF	ID	EX	MEM	HW	WB
LOAD	X	X				X
FETCH	X	X		X		X
STORE	x	x		x		

Eingabe- und Ausgabebefehle

Befehle	IF	ID	EX	MEM	HW	WB
INPUT	X	X			X	X
OUTPUT	X	X			X	

Logische, Arithmetische und Schiebebefehle

Befehle	IF	ID	EX	MEM	HW	WB
AND, OR, XOR,ADD, ADDCY, SUB, SUBCY,SL0, SL1, SLA, SLX, SR0, SR1, SRA, SRX, RL, RR	X	X	X			X
OUTPUT	X	X	X			

Sprungbefehle, ((Unterprogramm-undInterrupt- Steuerbefehle))

Befehle	IF	ID	EX	MEM	HW	WB
JUMP, CALL, RETURN, (RETURNI, ENABLE INTERRUPT, DISABLE INTERRUPT)	X	X				

3. Beschreibung der Prozessor-Komponenten

3.1 Instruction Fetch Unit

Diese Komponente ist dafür zuständig die Befehle aus dem Befehlsspeicher zu laden und beinhaltet zugleich auch den Befehlszähler.

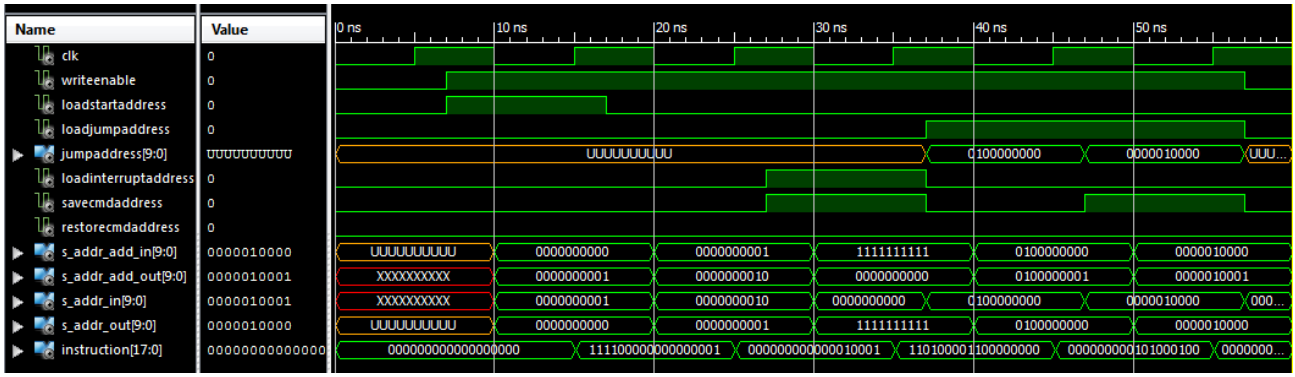
Der Befehlsspeicher wurde mittels dem KPCSM3 Assembler erstellt. Das Programm, welches letzendlich in unserem Gesamttest des Prozessors ausgeführt wurde, arbeitet die folgenden Befehle ab:

```
                ENABLE INTERRUPT
                LOAD s0, 11
                LOAD s1, 22
                ADD s1, s0
                STORE s1, 10
loop:   SL1 s0
                JUMP NC, loop
                INPUT s3, (s1)
                FETCH s4, 10
                OR s3, s4
                OUTPUT s3, 01
                ADDRESS 100
up:             LOAD s1, 44
                RETURN

                ADDRESS 300
isr:    CALL up
                LOAD s1, 55
                RETURNI ENABLE

                ADDRESS 3FF
                JUMP isr
```

Der Befehlsspeicher ist als Schnittstelle in der Instruction Fetch Unit eingebunden und es wird darauf mit einem 10 Bit breiten Signalvektor auf die Adresse des zu lesenden Befehls zugegriffen. Dadurch wird dann der gelesene Befehl in einem 18 Bit breiten Signalvektor vom PROM ausgegeben.

Der Befehlsspeicher hat eine automatische Verzögerung, da dieser aus der Bibliothek eingebunden wurde. Die Verzögerungszeiten wurden deshalb hier bereits automatisch angepasst.

Test der Instruction Fetch Unit



3.2 Instruction Decode Unit

Die Einheit zum Dekodieren des Befehls ist zuständig dafür den Befehl in seine einzelnen Funktionen aufzuteilen.

Aufbau eines Befehlswortes

- der Befehlscode ist über die Bits 17 – 13 verteilt
- der erste Operand der Befehle ist von Bit 11 – 8
- der zweite Operand adressiert entweder ein Register (Bit 7 - 4) oder aber eine Konstante die direkt weitergeleitet wird (Bits 7 - 0)

Funktionscode für Schiebe- und Rotationsbefehle
- das Richtungs-Bit (Bit 3) signalisiert die Richtung des Shift-Befehls
- der Funktionscode liegt hierbei an den Bits 2 und 1 an
- das Steuerbit ist auf Bit 0

Wenn ein Sprung abgearbeitet werden soll, so liegt die Sprungadresse an den Bits 9 bis 0 an. Die Sprungbedingung liegt auf den Bits 11 und 10 an.

Das Bit 12 signalisiert von welcher Art der zweite Operand ist: eine 0 bedeutet es ist eine Konstante und eine 1 bedeutet es ist ein Register, welches adressiert wird. Wenn es sich um einen Sprung handelt dann wird in diesem Bit hier mit 0 ein unbedingter Sprung oder mit 1 ein bedingter Sprung ausgedrückt. Die Auswahl zwischen den beiden Operanden wird durch einen 2-zu-1 Multiplexer durchgeführt.

Registersatz

Das Registerfile, dass die prozessorinternen Register beinhaltet ist ebenfalls Teil der Instruction Decode Unit. Dieser interne Speicher wird in unserem Programmablauf verwendet, da er schnelleren Zugriff auf Daten ermöglicht im Vergleich zur Verwendung eines externen Speichers.

Das Modul zum Speichern des Ergebnisses implementiert das Zurückschreiben in eines der vorhandenen Register.

Der Registersatz ermöglicht es 16 Datenwerte mit einer Breite von je 8 Bit abzuspeichern. Die Adressen, über welche die Datenwerte adressiert werden haben eine Breite von 4 Bit.

In unserer Implementation wurde hierfür ein verteilter Dual Port RAM verwendet, da zwei Operanden gelesen werden können müssen. Die Lese- und Schreibzugriffe erfolgen hierbei hintereinander.
Dieser Art von Speicher hat die Eigenschaft im Gegensatz zu einem Block-RAM weniger Fläche auf dem FPGA zu benötigen.

Das Speichern des Ergebnisses ist nur möglich wenn das Write Enable signal auf 1 gesetzt wird. Dieses Signal wird von der Control Unit gesteuert und an das Registerfile ausgegeben.

Test der Instruction Decode Unit

Beim Test der Instruction Decode Unit ist ersichtlich, dass diese Komponente nicht wie gewünscht funktioniert. Die Datenausgänge des 1. und 2. Leseports weisen hier nicht die richtigen Signale auf und aus Zeitgründen konnte dieser Fehler nicht behoben werden.

3.3 Control Unit

In der Steuer-Einheit werden die Signale zur Anpassung von Ein- und Ausgabeoperationen, Schreibfreigabesignale und Steuersignale für Multiplexer im Datenpfad erzeugt.
Außerdem werden hier die Bedingungen bei Sprungbefehlen getestet und hier werden auch Unterbrechungen im Programmablauf erkannt, die entweder durch Unterprogrammaufrufe oder Interrupts erzeugt werden.

Die Steuer-Einheit gibt außerdem das Reset-Signal an die weiteren Komponenten weiter. Der Befehlszähler (Laden der Startadresse) und das Flag-Register (Löschen des Flags) werden durch das Reset-Signal gesteuert. Wenn das Reset-Signal aktiv ist werden nur die gesteuerten Komponenten direkt beeinflusst. Der Systemzustand darf sich währenddessen nicht ändern. Aus diesem Grund müssen die Schreibfreigabesignale und das Signal zur Verlängerung von Hardwarezugriffen deaktiviert werden, da diese Zustandsänderungen steuern.

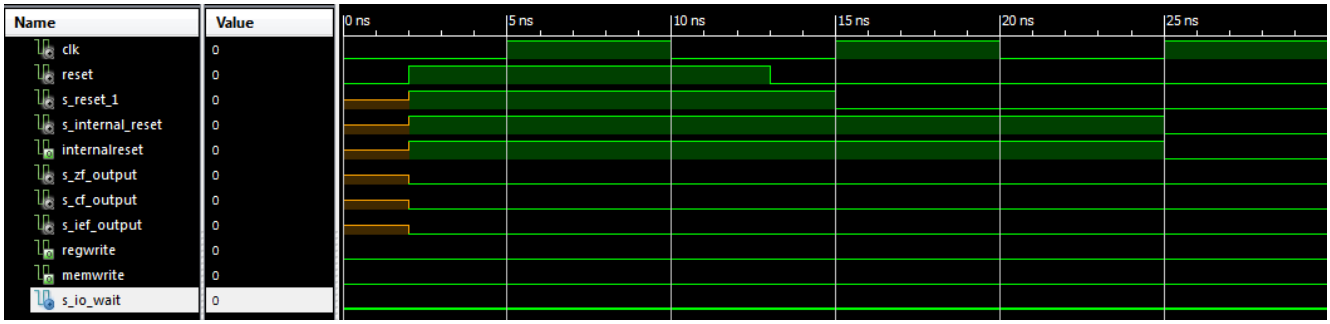
Beim lesende Zugriff auf externe Komponenten mittels dem INPUT-Befehl (dauert 2 Takte) wird das Ergebnis der Befehlsausführung am Ende der Taktperiode gespeichert. Erst im zweiten Takt erfolgt dann die Schreibfreigabe des Registersatzes.

Flagregister

Das Flagregister besteht im Grunde genommen aus dem 2-zu-1 Multiplexer und dem D-Flip-Flop mit Rücksetz- und Freigabeeingang. Das D-Flip-Flop wird als Speicher für 3 Bits eingesetzt. Diese 3 Bits repräsentieren das Carry-, Zero- und Interrupt-Enable-Flag. Der Einfluss eines Interrupts wurde zwar von uns in der Implementierung im Flagregister mit eingeführt, wird jedoch in der späteren Implementierung aus Zeitgründen nicht weiter in den Prozessor einbezogen.
Die Schreibfreigabe der Flip-Flops für die Speicherung des Carry- und Zero-Flag-Bits wird über das Write Enable-Signal gesteuert.

Test der Control Unit

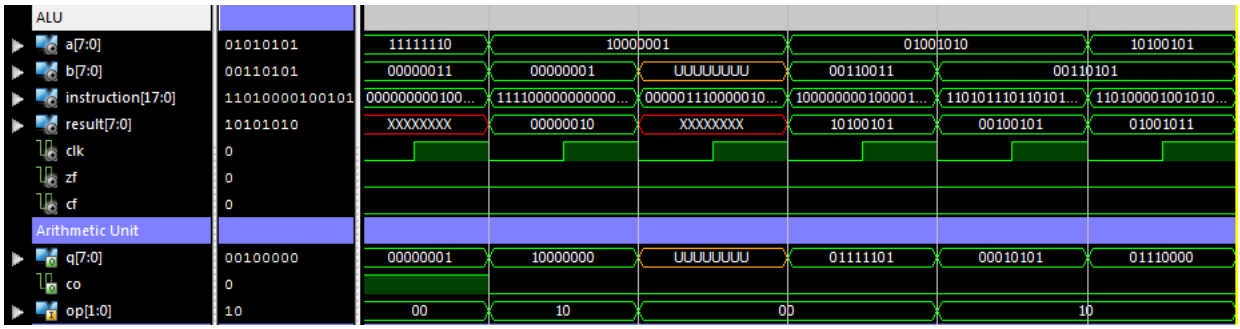
Rücksetzen des Prozessors



3.4 Arithmetic Logic Unit

Die Recheneinheit dient als Top-Modul für das Rechenwerk und das Flag-Register.

Test der ALU



3.4.1 Calculation Unit

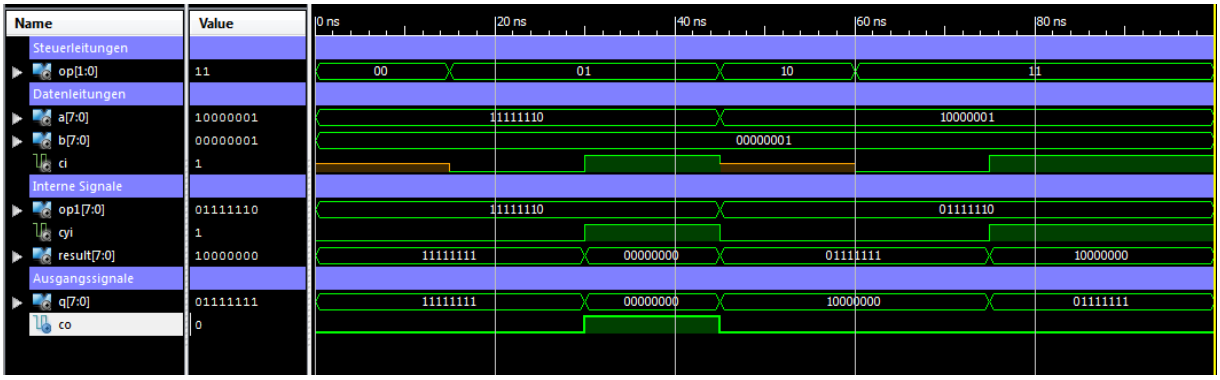
Vom Rechenwerk werden die Ergebnisse der arithmetischen, logischen und Schiebe- und Rotationsoperationen ausgegeben und das Zero-Flag berechnet.

3.4.1.1 Arithmetic Unit

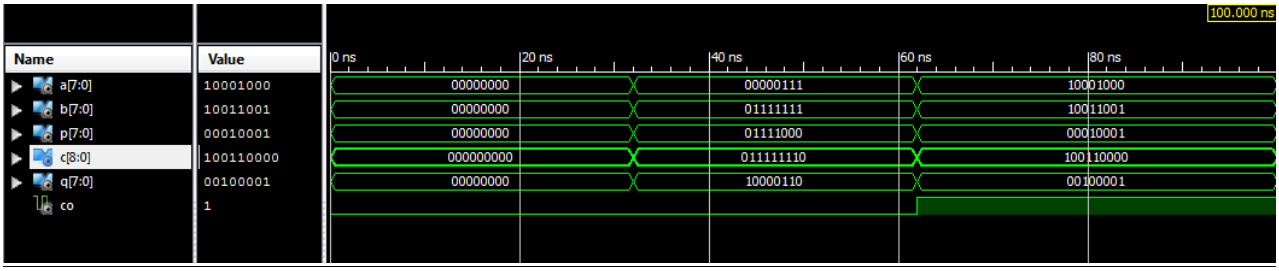
In der arithmetischen Einheit wurde von uns nur ein Full Adder eingefügt.. Dieser Addierer ist ein Carry-Chain-Addierer was eine spezielle Umsetzungsform eines Voll-Addierers ist und ist fähig die Befehle ADD, ADDCY, SUB, COMPARE und SUBCY auszuführen.

Diese Umsetzung ist eine platzsparende und zeitlich effiziente Umsetzung, ist jedoch komplexer als die algorithmische Umsetzung.

Test der Arithmetic Unit



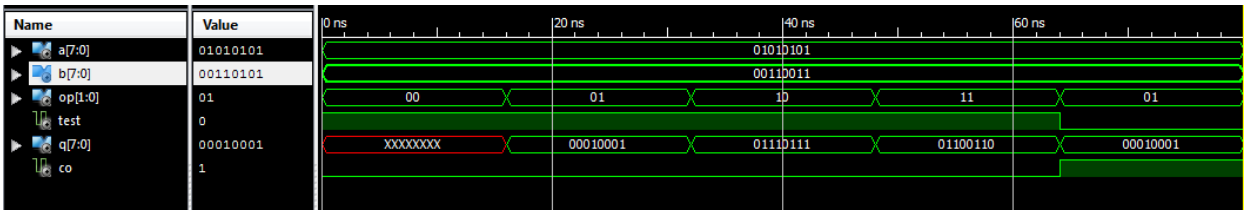
Test des Volladdierers



3.4.1.2 Logic Unit

Die logischen Operationen AND, Test, OR und XOR werden mittels den Bits 13 und 14 des Befehlscodes gesteuert. Die Operationen wurden direkt in VHDL umgesetzt. Es gibt hierbei nur zwei binär kodierte Steuersignale zur Unterscheidung der Operationen.

Test der Logic Unit



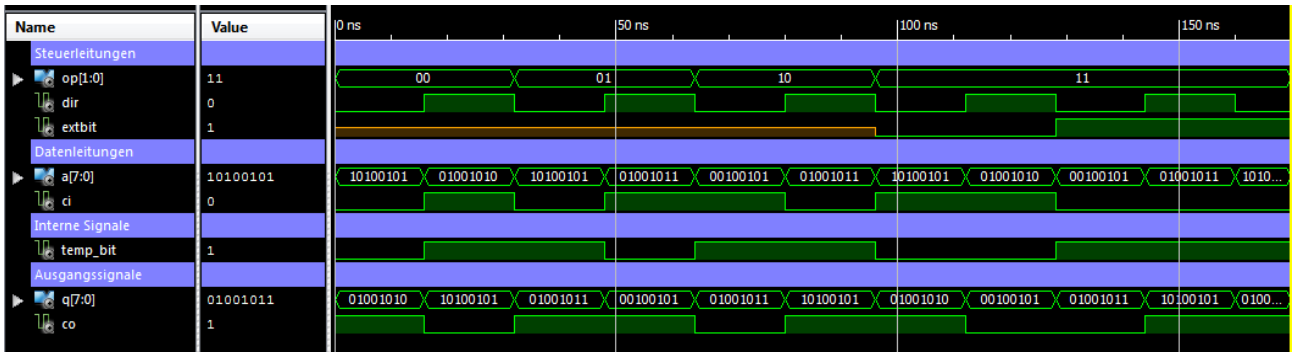
3.4.1.3 Shift Unit

Bei den Schiebe- und Rotationsoperationen wird die Operation durch das Bit 1 und 2 des Befehlscodes signalisiert und hat folgende Befehle:
- 00 – SLA, SRA
- 01 – RL, SRX
- 10 – SLX, RR
- 11 – SL0, SL1, SR0, SR1

Die Richtung wird vom Bit 3 des Befehlscodes angegeben, eine 0 bedeutet links und eine 1 bedeutet rechts.
Das Steuer-Bit im Bit 0 des Befehlscodes signalisiert mit 0 ein SL0, SR0 und mit 1 ein SL1, SR1.

Die Bildung des Carry-Flags ist nur von der Bewegungsrichtung abhängig.

Test der Shift Unit



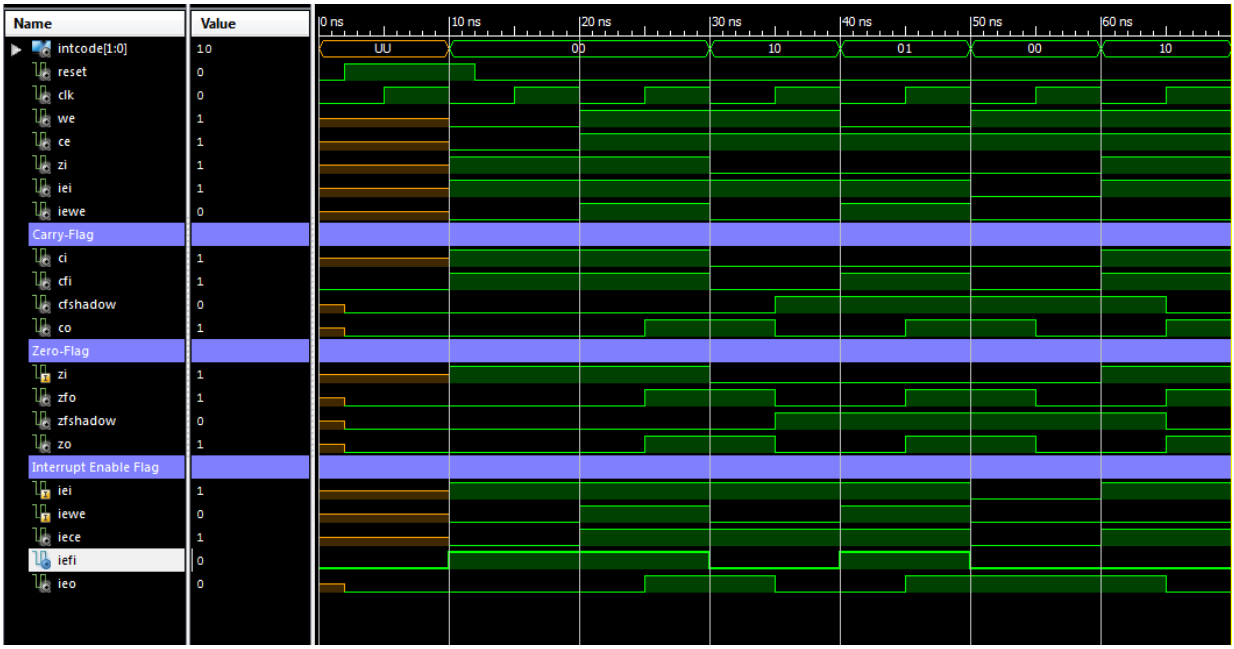
3.4.1.3 Zero Flag Calculation

Das Zero Flag wird gesetzt, wenn alle Bits des Ergebnisses aus dem Rechenwerk 0 sind.

3.4.2 Flag Register

Hier wird das gleiche Flag Register verwendet, wie es schon in der Control Unit vorhanden ist.

Test des Flag Registers



3.5 Data Memory Unit

Die Speicherung der 64 möglichen Datenwerte mit einer Breite von 8 Bit wurde als Single Port RAM erstellt. Hierbei ist nur ein Zugriff möglich, entweder lesen (FETCH) oder schreiben (STORE).

Der Speicher ist als verteilter Speicher aufgebaut und wird über einen 6 Bit langen Adresswert adressiert. Der Schreibzugriff erfolgt bei diesem RAM synchron und der Lesezugriff erfolgt asynchron.

Das Write Enable-Signal wird in der Datenspeicher-Einheit nur dann aktiviert, wenn die Schreibfreigabe durch den STORE-Befehl gegeben wurde.

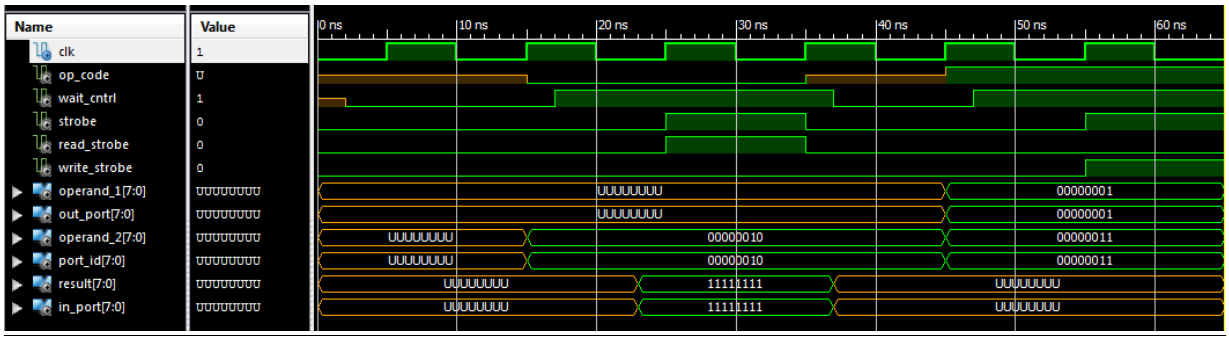
3.6 Hardware Access Unit

Die Ausführung der Ein- (INPUT) und Ausgabebefehle (OUTPUT) erfolgt über den Hardwarezugriff über 3 Ports mit einer Breite von je 8 Bit. Diese 3 Ports sind Dateneingang (IN_PORT), Datenausgang (OUT_PORT) und Adresse (PORT_ID).

Es ist hier ein weiteres D-Flip-Flop vorhanden, welches zur Lese- und Schreibsteuerung der externen Hardwarezugriffs-Einheit dient.

Das Timing des Schreibens in das Register geschieht hier beim Lesen sowie auch beim Schreiben immer mit der steigenden Flanke am Ende des Strobe-Signals. So wird der Wert des Dateneingangs mit der steigenden Flanke gelesen, während das READ_STROBE Signal aktiv ist und das Datenwort des Datenausgangs in ein Register geschrieben, wenn die steigende Flanke erfolgt und das WRITE_STROBE Signal aktiv ist.

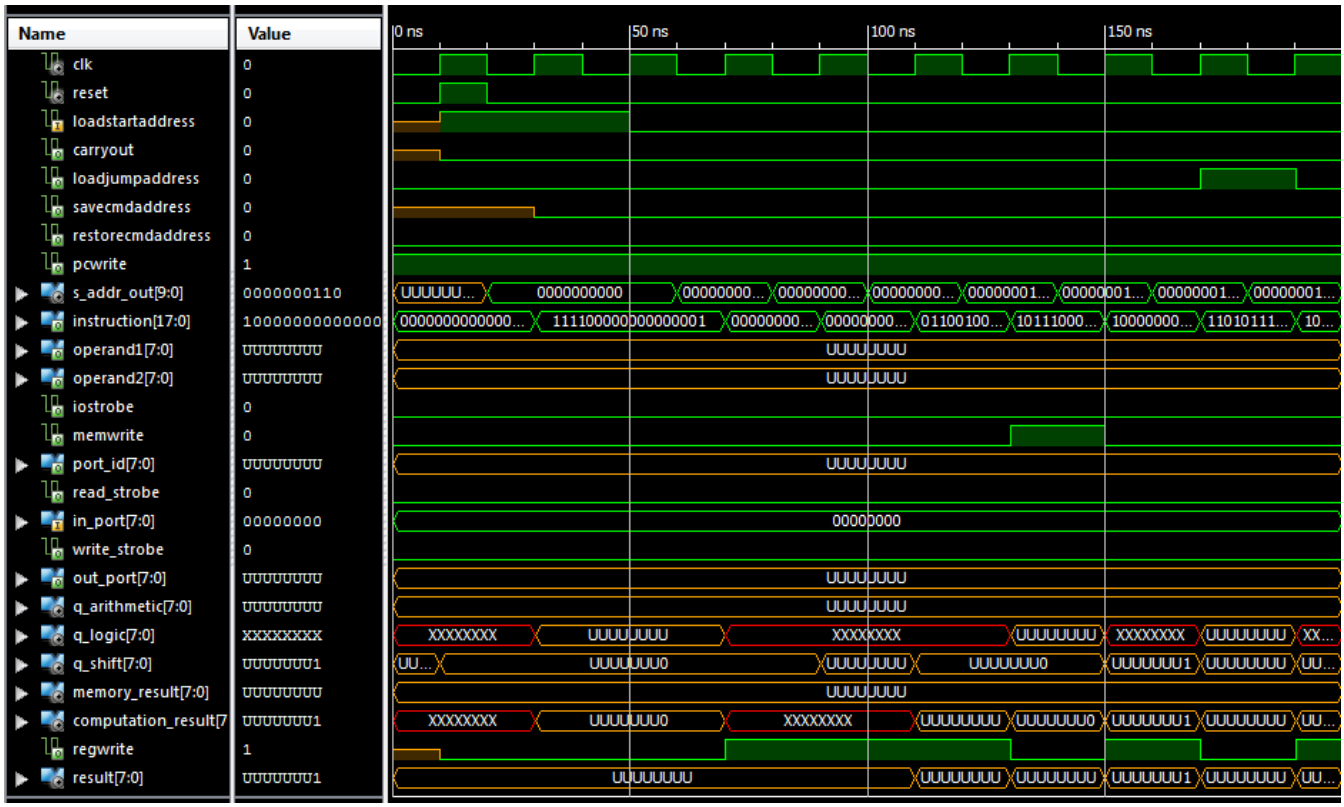
Test der Hardware Access Unit



3.7 Data Path Unit

Der Datenpfad-Umschalter wurde von uns direkt in den Prozessor-Komponente eingebaut. Er beinhaltet 3 Demultiplexer die den Datenpfad der Ergebnisse der Berechnungseinheit, der Datenspeicherzugriffe, Hardwarezugriffe und der Umgehung umschaltet.

4. Test des Prozessors



Der Test unseres Prozessors ist auf Grund eines Fehlers in der Instruction Decode Unit nicht erfolgreich verlaufen. Auf diesen Fehler sind wir bereits in dem Test der Komponente weiter eingegangen.

Alle anderen Komponenten des Prozessors funktionieren wie gewünscht.