Hausaufgabe 4: PicoBlaze Assembler

P. Gänz, L. Wagner

36 Punkte

Aufgabenstellung

Schreiben Sie Programme in PicoBlaze, welche die folgenden Probleme zu lösen. Ihre Abgabe sollten den Quellcode ihrer Programme enthalten.

Allgemeine Bearbeitungshinweise:
Werte deren Basis nicht explizit angegeben sind haben die Basis 10. Benutzen Sie so weit wie möglich sinnvolle Schleifenkonstrukte. Kommentieren Sie Ihre geplante Vorgehensweise im Code (';' startet einen einzeiligen Kommentar). In Aufgabe 3 und 4 werden Werte mittels Zweierkomplementdarstellung interpretiert. Benutzen Sie bitte ausschließlich den Befehlssatz von PicoBlaze3 (dieser wurde in der Vorlesung vorgestellt).

Abgabehinweise:
Bitte geben Sie Ihre Lösung der jeweiligen Aufgabe als einzelne Datei ab. Benennen Sie Ihre abgegebenen Dateien bitte nach dem Schema HA1.psm, HA2.psm,... abhängig von der Aufgabennummer.

Allgemeine Hinweise zu PicoBlaze:
PicoBlaze ist eine Assembler Programmiersprache für FPGAs und CPLDs des Unternehmens Xilinx. Das heißt, dass in PicoBlaze geschriebene Programme nicht direkt auf Ihrem x86 oder ARM/M1-basierten Prozessoren ausführbar sind. Sie können allerdings mit Tools wie dem Open PicoBlaze Assembler/Simulator[1] (Benötigt Python/Nim) oder KPicoSim[2] (Benötigt Ubuntu 11 + KDE 3.5) simuliert werden.

1 Zählen der mit Eins belegten Bits (6 Punkte)

Zählen Sie die Anzahl der ungeraden Bits (Einsen) eines Wertes in Binärdarstellung. Gehen Sie davon aus, dass Register s0 die Eingabe enthält. Das Ergebnis soll in Register sF abgelegt werden.

Beispiele:

$01110101_2 \Rightarrow 5_{16}$
$01100010_2 \Rightarrow 3_{16}$

[1] `http://kevinpt.github.io/opbasm/`
[2] `https://github.com/thomas-scheiblhuber/KPicoSim`

2 Johnson-Zähler Simulation (6 Punkte)

Ein Johnson-Zähler ist eine Variation eines Schieberegisters, bei dem der letzte Datenausgang (LSB) negiert auf den ersten Dateneingang (MSB) zurückgeführt wird.
Ihre Aufgabe ist es, einen nach rechts schiebenden 8-Bit Johnson-Zähler zu simulieren. Der Initialwert des Zählers soll aus dem Scratchpad (RAM) von Adresse 0 gelesen werden. Schreiben Sie alle Werte, die der Zähler annehmen kann in das Scratchpad (RAM). Beginnen Sie dabei, ab Adresse 0.
Beispiel:
Input: s0=$1A_{16} = 0001\ 1010_2$
Output an der Registeradresse...

0: $1A_{16}$
1: $8D_{16}$
2: 46_{16}
⋮

3 Modulo Berechen (7 Punkte)

Berechnen Sie den Rest einer Division (Modulo) zweier positiver 16-Bit Zahlen. Gehen Sie davon aus, dass Zahl 1 in den Registern s0 und s1 sowie Zahl 2 in den Registern s2 und s3 abgelegt sind. Wobei Register s0 und s2 die höher wertigen Bits enthalten. Der berechnete Rest muss in Registern sE (höherwertige Bits) und sF (niedrigwertige Bits) abgelegt werden.
Beispiel:
Input: s0=01_{16}, s2=00_{16}, s3=00_{16}, s4=22_{16}
Output: sE=00_{16},sF=12_{16}

4 Multiplikation mit einer Zweierpotenz (8 Punkte)

Realisieren Sie die Multiplikation eines 8-Bit Wertes x mit einer Zweier-Potenz einer positiven 8-Bit Zahl n ($f(x) = x*(2^n)$), verwenden Sie ein 16 Bit-Ergebnis. Gehen Sie davon aus, dass x und n an den Adressen 0 und 1 des Scratchpads (RAM) gespeichert sind. Ihr Ergebnis soll in den Registern sE (höher wertige Bits) und sF (niedriger wertige Bits) abgelegt werden.
Beispiel:
Input: RAM[0]=11_{16}, RAM[1]=5_{16}
Output: sE=02_{16},sF=20_{16}

5 Mantissenmultiplikation und Normalisierung (9 Punkte)

In dieser Aufgabe sollen Sie Teile der Multiplikation von zwei 8-Bit Fließkommazahl realisieren. Sie können davon ausgehen, dass sich im Scratchpad an den Adressen 0 und 1 jeweils zwei normalisierte Fließkommazahlen befinden. Diese Zahlen haben eine Fließkommadarstellung mit 3 Exponenten-Bits und 4 Mantissen-Bits. Multiplizieren Sie die Mantissenwerte möglichst effizient miteinander. Speichern Sie die resultierenden Mantissenbits in Register sF. Verwenden Sie Runden in Richtung Null (Round towards Zero).
Beispiel:
Input: RAM[0]=35_{16}, RAM[1]=23_{16}
Output: sF=08_{16}