Rechnerarchitektur
Sommersemester 25

ertis embedded real-time systems ulm university institute

Laborprojekt – Versuch 3

In Ihrem letzten Versuch haben Sie die generischen Komponenten ALU, 2-1-Multiplexer und Pipeline-Register erstellt. In diesem Versuch erweitern wir die Komponenten um ein Registerfile für den RISC-V (ähnlich dem der MIPS aus GdTi), ein Befehls-Decoder (beschränkt auf die Registerbefehle) und der Sign-Extension. Die Sign-Extension wird für alle Befehlsarten ausgelegt und soll die möglichen Immediates gemäß Spezifikation auf 32-Bit erweitern.

Ziele

- Konzeptionierung, Umsetzung und Validierung eines Befehls-Decoders für R-Befehle
- Konzeptionierung, Umsetzung und Validierung einer RISC-V-Sign-Extension

Vorbereitung

Spezifikation RISC-V (R/I-Befehle)

Lesen Sie sich in die Register- und Immediate-Befehle in der Referenz (RV32I Base Integer Instruction Set, Version 2.0) ein und eignen Sie sich ein Grundverständnis der Befehlsstruktur so an, dass Sie die Befehlscodierung mithilfe der Referenzkarten wiedergeben und erklären können. Erarbeiten Sie sich ein Konzept zur Umsetzung der Sign-Extension aller notwendigen Immediates.

GHDL-Standards

Lesen Sie sich in die GHDL-Optionen, insbesondere wie Sie Standards der Sprache einstellen, ein. Ab sofort werden alle Komponenten und Testbenches nur noch mit VHDL 2008 genutzt.

Zusatzdateien

Erarbeiten Sie sich die Inhalte der Ihnen zur Verfügung gestellten Zusatzdateien so, dass Sie diese anhand des Codes erklären können.

Aufgaben

Aufgabe 1: Befehls-Decoder für R-Befehle

Ein Befehls-Decoder ist eine wichtige Komponente in Prozessoren wie dem RISC-V, die zur Zerlegung des Befehls, in Abhängigkeit der Art des Befehl, in Steuersignale dient. In diesem Versuch beschränkt sich der Befehls-Decoder auf die **R**egister-Befehle und wird in den kommenden Versuchen sukzessive erweitert. Das Verhalten des Befehls-Decoders besteht hauptsächlich aus zwei Phasen, dem Erkennen des Befehlstyps und der Ausgabe der Befehlsspezifischen Steuersignale.

Für die Abbildung der Steuersignale des Befehls-Decoders sollen Sie in diesem Versuch den sogenannten *record*-Typ nutzen. In der Datei *types.vhdl* ist dazu das *controlword* vom Typ *record* definiert und beinhaltet alle notwendigen Steuersignale (bereits für alle Befehlstypen). Zusätzlich ist die Konstante *control_word_init* vom Typ *record* definiert, mit der das *controlword* mit Nullen initialisiert werden kann.

Bei dem genutzten *controlword* ist für diesen Versuch nur das Steuersignal *ALU_OP* und *I_IMM_SEL* von Interesse. Das Steuersignal *ALU_OP* wird aus zwei Teilen des Befehlsworts zusammengesetzt, der sogenannten **funct7** und **funct3**, wobei von der **funct7** nur das zweite Bit genommen wird (siehe Spezifikation RISC-V). Dieses Bit zeigt an, ob es sich um eine Subtraktion handelt. Damit entspricht *ALU_OP* den 4-Bit-langen Werten der ALU-Operationen im *Constant_package*. Das Steuersignal *I_IMM_SEL* wird zu diesem Zeitpunkt fest auf 0 gesetzt.

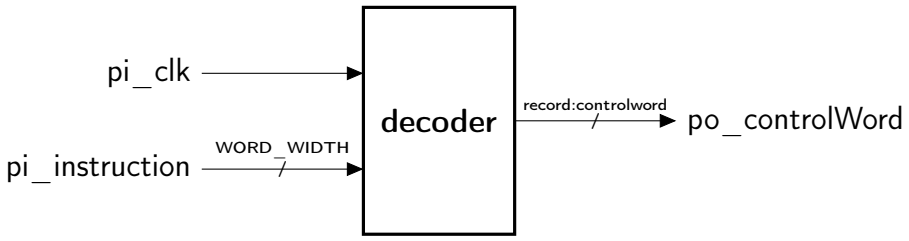

Abbildung 1: decoder

Bei einer steigenden Flanke des Taktsignals *pi_clk* soll die Eingabe am Eingang *pi_instruction* der Länge *WORD_WIDTH* als Ausgabe vom Typ *controlword* am Ausgang *po_controlWord* erscheinen. Die Umsetzung soll aus einem Takt-sensitiven Prozess bestehen, der in der ersten Phase über *cases* die Art des Befehls identifiziert. In diesem Versuch beschränken wir uns ausschließlich auf die R-Befehle. Wenn der OP-Code des Befehlsworts der Konstanten *R_OP_INS* entspricht, soll der Variable *v_insFormat* das *rFormat* aus dem *types.vhdl* zugeordnet werden.

In einer zweiten Case-Abfrage im selben Prozess, sollen nun die Steuersignale gemäß Spezifikation auf den Ausgang *po_controlWord* gesetzt werden.

- Erstellen Sie den Unterordner *<Projektordner>/Komponenten/Decoder*.
- Lesen Sie sich in die Funktionsweise und Zusammensetzung der R-Befehle ein und erklären Sie, basierend auf der RISC-V Spezifikation, diese in Kommentaren in der Datei *decoder.vhdl*.
- Erstellen Sie die Entity decoder in der Datei *decoder.vhdl* und setzen Sie das beschriebene Verhalten um.
- Überprüfen Sie ihre Umsetzung mit der Testbench *decoder_tb* in der Datei *decoder_tb.vhdl* im Unterordner *<Projektordner>/Testbenches/Decoder* erfolgreich.
- Erstellen Sie ein Bash-Skript im Ordner *<Projektordner>/Testbenches/Decoder*, mit dem Sie die Simulation der Decoder-Testbench inklusive der Analyse und Elaboration automatisch und erfolgreich ausführen können.

Abnahme Funktion des Decoders, Programmierkonventionen, RISC-V Spezifikation der Registerbefehle.

Aufgabe 2: Sign-Extension

In RISC-V ist die Sign-Extension wichtig, da die Befehlssätze normalerweise auf eine feste Breite von 32 Bit begrenzt sind, während die Operanden möglicherweise kürzer sind. Wenn also ein Wert, der in einem Befehl verwendet wird, kürzer als die Breite des Befehls ist, muss er auf die volle Breite des Befehls erweitert werden, damit das Vorzeichen korrekt behandelt wird.

Die Sign-Extension in RISC-V erfolgt grundsätzlich wie folgt:

Zunächst wird der Wert aus dem Speicher oder aus einem Register geladen. Anschließend wird das Vorzeichenbit auf die volle Breite des Zielregisters oder der Zieloperation erweitert. Dadurch bleibt das ursprüngliche Vorzeichen korrekt erhalten, positive Werte bleiben positiv und negative Werte negativ.

Der von Ihnen umzusetzende Sign-Extender soll über einen Eingang *pi_instr* in Breite der Befehle verfügen und alle fünf möglichen Immediate-Werte (*po_immediateImm*,, (*po_storeImm*, *po_unsignedImm*, *po_branchImm*, *po_jumpImm*) gemäß Spezifikation des RISC-V ausgeben.

- Erstellen Sie den Unterordner *<Projektordner>/Komponenten/SignExtender*.
- Erstellen Sie die Entity signExtension in der Datei *signExtension.vhdl* und setzen Sie das beschriebene Verhalten um.

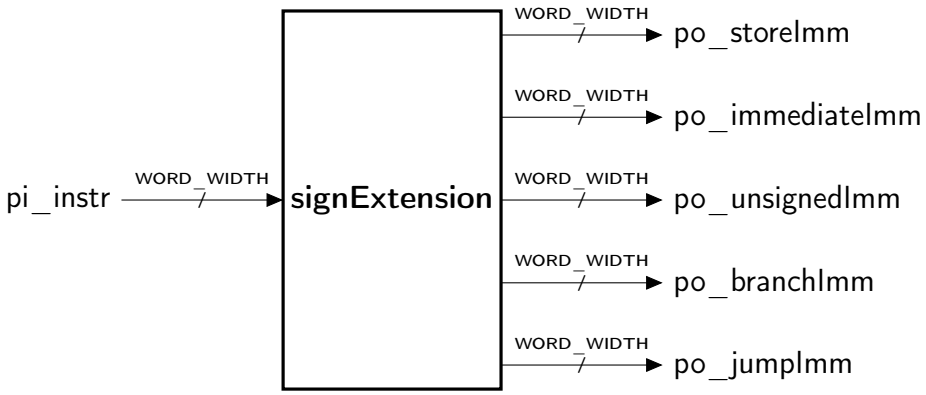

Abbildung 2: signExtension

- Überprüfen Sie ihre Umsetzung mit der Testbench *signExtension_tb* in der Datei *signExtension_tb.vhdl* im Unterordner *<Projektordner>/Testbenches/SignExtender* erfolgreich.
- Erstellen Sie ein Bash-Skript im Ordner *<Projektordner>/Skripts/SignExtender*, mit dem Sie die Simulation der Sign-Extender-Testbench inklusive der Analyse und Elaboration automatisch und erfolgreich ausführen können.

Abnahme *Funktion der Sign-Extension*, Programmierkonventionen, RISC-V Spezifikation der Registerbefehle.

Aufgabe 3: Abgabe

Laden Sie die erstellte Ordnerstruktur mit den dazugehörigen Dateien und einer Anleitung in Form (*readme_v3.md*) als Zip-Datei unter dem Namen

Name_Vorname_Versuch3.zip in Moodle hoch.