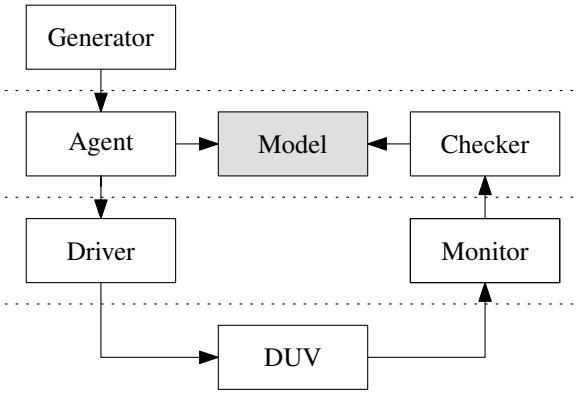
FH-OÖ Hagenberg/ESD
Advanced Methods of Verification, SS 2020
Rainer Findenig

3. Übung: Behavioral Modeling

1 The Bigger Picture

In den nächsten vier Übungen werden Sie eine komplette Verifikationsumgebung [Spe06] für den PROL16 entwickeln. Einen Überblick über den Aufbau finden Sie in der folgenen Abbildung:



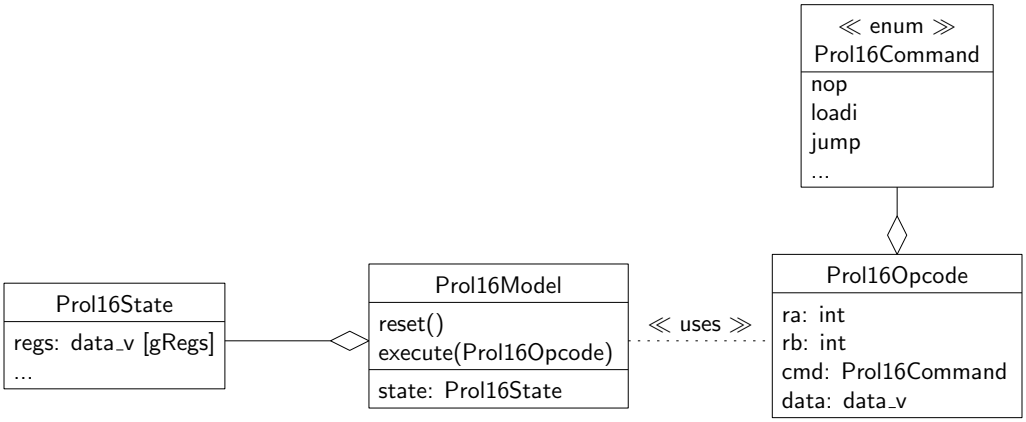
2 Behavioral-Modellierung der PROL16-Architektur

Eine weit verbreitete Vorgehensweise im Systementwurf ist die Verwendung von Modellen auf einer möglichst hohen Abstraktionsebene (*Transaction Level Modeling*). Dies bietet mehrere Vorteile:

- Eine höhere Abstraktionsebene ermöglicht eine „Entwurfsraumexploration“, bei der man verschiedene mögliche Architekturen gegenüberstellt und eine (möglichst) ideale Alternative ausgewählt.
- Mit steigender Abstraktion sinkt in der Regel der Implementierungsaufwand und damit zumindest tendenziell auch die Anzahl der Fehler.
- Ein abstraktes Modell ist meist nicht *cycle accurate*, sondern „rechnet“ ohne (*untimed* bzw. *programmer's view*) oder nur mit geschätzter Verzögerung (*approximately timed* bzw. *programmer's view with timing*). Dies kann die Implementierung erheblich vereinfachen und die Simulation beschleunigen.

- Im Idealfall kann das Modell in weiteren Verfeinerungsschritten auch als Referenzmodell verwendet werden.

In dieser Übung soll nun ein solches Verhaltensmodell der PROL16-Architektur in SystemVerilog erstellt werden. Implementieren Sie dazu die folgende Klassenstruktur:



Bei jedem Aufruf der Funktion `Prol16Model::execute()` soll genau ein `Prol16Opcode` abgearbeitet werden. Beachten Sie, dass das Modell nicht *cycle accurate* sein soll, diese Berechnung also *sofort* (ohne Zeitverzögerung) erfolgen soll[1].

Der Befehlssatz des PROL16 ist in der folgenden Tabelle zusammengefasst [KSL03]:

Binary	Dez.	Mnemonic	Op1	Op2	Word(s)	Cycles	Description	C-Flag	Z-Flag
000000	0	**NOP**	-	-	1	2	-		
000001	1	**SLEEP**	-	-	1	2	stop simulation		
000010	2	**LOADI**	Ra	immediate	2	3	Ra:=immediate		
000011	3	**LOAD**	Ra	Rb=Addr	1	3	Ra:=Mem(Rb)		
000100	4	**STORE**	Ra	Rb=Addr	1	3	Mem(Rb):=Ra		
001000	8	**JUMP**	Ra=Addr	-	1	2	PC:=Ra		
001010	10	**JUMPC**	Ra=Addr	-	1	2	PC:=Ra if (C=1)		
001011	11	**JUMPZ**	Ra=Addr	-	1	2	PC:=Ra if (Z=1)		
001100	12	**MOVE**	Ra	Rb	1	2	Ra:=Rb		
010000	16	**AND**	Ra	Rb	1	2	Ra:=Ra and Rb	0	x
010001	17	**OR**	Ra	Rb	1	2	Ra:=Ra or Rb	0	x
010010	18	**XOR**	Ra	Rb	1	2	Ra:=Ra xor Rb	0	x
010011	19	**NOT**	Ra		1	2	Ra:=not(Ra)	0	x
010100	20	**ADD**	Ra	Rb	1	2	Ra:=Ra+Rb	x	x
010101	21	**ADDC**	Ra	Rb	1	2	Ra:=Ra+Rb+Carry	x	x
010110	22	**SUB**	Ra	Rb	1	2	Ra:=Ra-Rb	x	x
010111	23	**SUBC**	Ra	Rb	1	2	Ra:=Ra-Rb-Carry	x	x
011000	24	**COMP**	Ra	Rb	1	2	Ra-Rb	x	x
011010	26	**INC**	Ra	-	1	2	Ra:=Ra+1	x	x
011011	27	**DEC**	Ra	-	1	2	Ra:=Ra-1	x	x
011100	28	**SHL**	Ra	-	1	2	Ra:=Ra << 1	x	x
011101	29	**SHR**	Ra	-	1	2	Ra:=Ra >> 1	x	x
011110	30	**SHLC**	Ra	-	1	2	Ra:=Ra << 1 (with Carry)	x	x
011111	31	**SHRC**	Ra	-	1	2	Ra:=Ra >> 1 (with Carry)	x	x

[1] Das Modell hat auch keinen Takteingang!

Beachten Sie die folgenden Punkte:

- Erstellen Sie eine Testbench, die eine Instanz der Klasse `Prol16Model` erzeugt und Befehle auf diesem Modell ausführt. Erweitern Sie das Modell gegebenenfalls um eine Funktion zur Ausgabe des aktuellen Status des Modells, um die Ausführung überprüfen zu können.
- Sie müssen kein externes RAM implementieren.
- Das Modell soll nur den *internen* Zustand (Register, Program Counter, Flags) des PROL16 umfassen, eine Speicherschnittstelle oder ähnliches muss nicht implementiert werden.
- Die Befehle **SLEEP**, **LOAD** und **STORE** müssen von dem Modell nicht unterstützt werden.
- Ungültige Befehle (nicht definierte Opcodes, nicht existente Register) müssen als **NOP** behandelt werden.
- Beachten Sie, dass in den folgenden Übungen eine die Klasse `Prol16Opcode` so erweitert werden muss, dass Instanzen in den entsprechenden Bitvektor umgerechnet werden können.
- Die Variable `data` der Klasse `Prol16Opcode` wird nur für den Befehl **LOADI** verwendet.

"Oh behave!"
Austin Powers

Literatur

[KSL03] Thomas Klaus, Markus Schutti, and Markus Lindorfer. *Befehlssatz PROL16*. Institut für Integrierte Schaltungen, Johannes Kepler Universität Linz, 2003.

[Spe06] Chris Spear. *SystemVerilog for Verification*. Springer Science+Business Media, 2006.