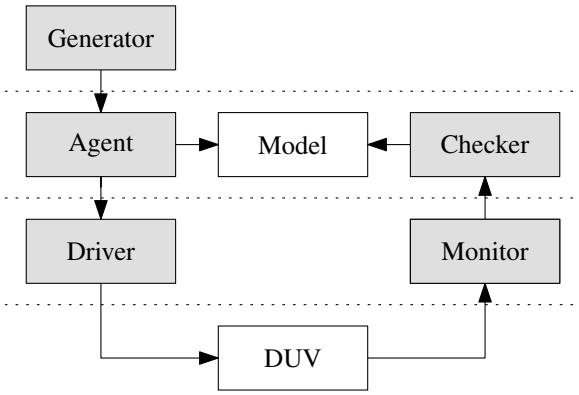
1 Verifikation des PROL16

In dieser Übung sollen Sie Ihren PROL16-Entwurf aus dem fünften Semester mit Hilfe des in der letzten Übung erstellten Modells (*"Golden Model"*) verifizieren. Dazu soll jeder generierte Befehl (also jede Instanz der Klasse `Prol16Opcode`) nicht nur von dem Modell, sondern auch auf dem VHDL-Entwurf exekutiert werden. Nach jedem Befehl kann der Registerinhalt der beiden Beschreibungen verglichen und damit geprüft werden, ob ein Fehler in der Befehlsausführung aufgetreten ist.

Die Testbench ist also wie im Folgenden dargestellt aufgebaut:



Erweitern Sie Ihre Testbench also aus der letzten Übung also so, dass das VHDL-Modell des PROL16 instantiiert und mit den Befehlen des Stimulus-Generators gespeist wird. Dabei wird ein Befehl nach dem anderen an den PROL16 angelegt, die vom Prozessor vorgegebene Adresse muss *nicht* beachtet werden! Die Befehle **SLEEP**, **LOAD** und **STORE** werden von dem Modell nicht unterstützt und daher in dieser Simulation vernachlässigt.

1.1 Zugriff auf VHDL-Signale aus der Testbench

Um den Vergleich des Zustands des Modells mit dem des VHDL-Entwurfs zu ermöglichen, muss aus der SystemVerilog-Testbench auf Signale des VHDL-Entwurfs zugegriffen werden. Da dies von VHDL nicht unterstützt wird, müssen dazu simulatorspezifische Funktionen verwendet werden. So können Sie mit *QuestaSim* auf ein Signal `pc`, das in der Instanz `/top/duv/datapath_inst` existiert, aus der Testbench folgendermaßen zugreifen:

```
module top;
    ...
    Prol16Test tb(...);
    cpu duv(...);
endmodule


program Prol16Test(...);
  logic [15:0] cpu_pc;
  ...
  initial begin
    $init_signal_spy("/top/duv/datapath_inst/pc", "/top/tb/cpu_pc");
    ...
    assert (cpu_pc == 0);
  end
endprogram
```

Beachten Sie, dass die *SignalSpy*-Funktion von *QuestaSim* keine RAMs (zweidimensionale Arrays) unterstützt – Sie müssen also jedes Register der Registerbank einzeln in die Testbench einbinden.

1.2 Reset

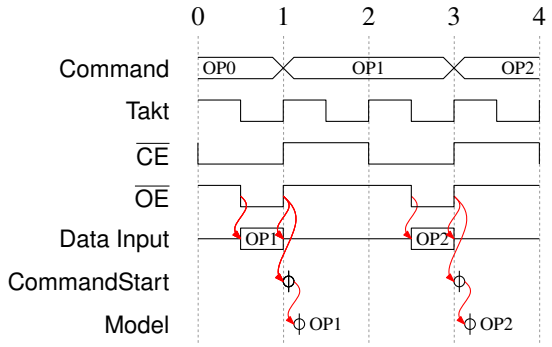
Das Register des PROL16 werden beim Reset nicht zurückgesetzt, haben also zu Beginn der Simulation keinen definierten Wert. Mit simulatorspezifischen Funktionen können Sie einen definitieren Initialwert setzen (Beispiel *QuestaSim*):

```
program Prol16Test(...);
  initial begin
    $signal_force("/top/duv/datapath_inst/thereg_file/registers(0)",
            "16#0000", 0, 1);
    ...
  end
endprogram
```

□ Erklären Sie die Parameter dieser Funktion!

1.3 Synchronisation

Die Herausforderung an dieser Testbench liegt in der Synchronistation der Testbench, des DUVs und des Modells. Eine einfache Lösung kann dabei mit *Events* realisiert werden, wie im Folgenden dargestellt:



- Mit der fallenden Flanke von $\overline{\text{OE}}$ muss der *nächste* Befehl (OP1) an das DUV angelegt werden.
- Mit der steigenden Taktflanke wird der Befehl in der CPU übernommen und gleichzeitig die Ergebnisse des *letzten* Befehls (OP0) in die Register geschrieben. Direkt danach wird $\overline{\text{OE}}$ wieder auf *high* gesetzt.
- Es ist also sinnvoll, direkt nach der steigenden Taktflanke ein Event (CommandStart) zu generieren, aufgrund dessen der Registerinhalt (Ergebnis von OP0) überprüft wird. *Nach* dieser Überprüfung kann das Modell den neuen Befehl (OP1) ausführen (Event Model).
- Mit der nächsten fallenden Flanke von $\overline{\text{OE}}$ wird wieder beim ersten Punkt begonnen. Die einzige Ausnahme bildet der Befehl **LOADI**, bei dem zu diesem Zeitpunkt die Daten angelegt werden müssen, und erst bei der nächsten fallenden Flanke wieder von vorne begonnen wird.

"There has never been an unexpectedly short debugging period in the history of computers."

Steven Levy