Übung zur Vorlesung Einführung in Computer Microsystems

Prof. Dr-Ing. A. Koch
Jaco Hofmann, MSc.

TECHNISCHE UNIVERSITÄT DARMSTADT

Sommersemester 15
Übungsblatt 3

Aufgabe 3.1 Scheduling

Zu Konflikten kommt es in Bluespec unter anderem wenn mehrere Rules versuchen das selbe Register zu schreiben. In dieser Aufgabe wird eine Möglichkeit gezeigt diese Konflikte gezielt zu lösen.

Aufgabe 3.1.1 Das Design

Zur Demonstration von Scheduling-Attributen wird folgendes Interface verwendet.

```
interface Conflict;
  method Action putA(UInt#(32) val);
  method Action putB(UInt#(32) val);
  method UInt#(32) getVal();
endinterface
```

Intern sollen beide Actions in das selbe Register schreiben.
Eine mögliche Implementation mit der Hilfe von `FIFO` sieht folgendermaßen aus.

```
 import FIFO::*;

 module mkConflict(Conflict);
  FIFO#(UInt#(32)) valA <- mkFIFO();
  FIFO#(UInt#(32)) valB <- mkFIFO();

  Reg#(UInt#(32)) superVal <- mkReg(0);

  rule writeA;
    let v = valA.first(); valA.deq();
    superVal <= v;
  endrule

  rule writeB;
    let v = valB.first(); valB.deq();
    superVal <= v;
  endrule

  method Action putA(UInt#(32) val);
    valA.enq(val);
  endmethod

  method Action putB(UInt#(32) val);
    valB.enq(val);
  endmethod

  method UInt#(32) getVal();
```

```
    return superVal;
  endmethod
endmodule

module mkConflictTestbench(Empty);
  Conflict uut           <- mkConflict();
  Reg#(UInt#(7)) counter <- mkReg(0);

  rule spamConflictA (pack(counter)[0] == 1);
    uut.putA(42);
  endrule

  rule spamConflictB;
    uut.putB(125);
  endrule

  rule cntrIncr;
    counter <= counter + 1;
  endrule

  rule endSim (counter == 127);
    $finish();
  endrule
endmodule
```

Beim Versuch oben stehenden Code zu kompilieren wird eine Warnung ausgegeben. Welche Auswirkungen hat diese Warnung auf das erstellte Design? Untersuchen Sie die Waveforms der Simulation mit `-keep-fires`.

Aufgabe 3.1.2 Steuerung

Die Priorisierung der Rules kann in Bluespec unter anderem mit Hilfe von Attributen gesteuert werden. Setzen Sie das Attribut `(* preempts = "writeA, writeB" *)` vor eine der Rules des Moduls `mkConflict`. Analysieren Sie die nun entstandenen Waveforms. Tauschen Sie `writeA` und `writeB` im obigem Attribut und schauen Sie sich erneut die Waveforms an.

Aufgabe 3.1.3 Weitere Attribute

Bluespec bietet viele weitere Attribute zur feinen Steuerung der Ausführungsreihenfolge von Rules. Welche Auswirkungen hat es wenn Sie statt `preempts` die Attribute `descending_urgency` oder `excecution_order` verwenden?

Aufgabe 3.2 BRAM

Implementieren Sie auf Basis des Packages BRAM einen RAM mit vier Eingängen. Das zu verwendende Modul sieht dabei folgendermaßen aus:

```
module mkBRAM1Server #( BRAM_Configure cfg )
  ( BRAM1Port #(addr, data) )
  provisos(Bits#(addr, addr_sz),
  Bits#(data, data_sz),
  DefaultValue#(data) );
```

Das Interface ist folgendermaßen definiert:

```
typedef Server#(BRAMRequest#(addr, data), data) BRAMServer#(type addr, type data);
typedef struct {Bool write;
    Bool responseOnWrite;
    addr address;
    data datain;
```

```
  } BRAMRequest#(type addr, type data) deriving(Bits, Eq);
interface BRAM1Port#(type addr, type data);
  interface BRAMServer#(addr, data) portA;
  method Action portAClear;
endinterface: BRAM1Port
```

Eine Instanziierung könnte wie folgt aussehen:

```
BRAM_Configure cfg = defaultValue; //declare variable cfg
cfg.memorySize = 1024*32; //new value for memorySize
BRAM1Port#(UInt#(15), Bit#(16)) bram <- mkBRAM1Server (cfg);
```

Da das Interface `BRAM1Port` nur einen Port für den RAM zur Verfügung stellt müssen Sie den vorhandenen Port zwischen den angeschlossenen Geräten teilen. Erlauben Sie jedem der Eingänge nacheinander für einen Takt den BRAM zu benutzen.

Das Interface des Moduls soll dabei folgendermaßen aussehen:

```
interface RRRam;
  BRAMServer#(UInt#(15), Bit#(16)) portA;
  BRAMServer#(UInt#(15), Bit#(16)) portB;
  BRAMServer#(UInt#(15), Bit#(16)) portC;
  BRAMServer#(UInt#(15), Bit#(16)) portD;
endinterface
```

Erstellen Sie eine Testbench die die vier Eingänge des BRAM mit Testdaten versieht. PortA soll dabei jeden Takt geschrieben werden, PortB jeden zweiten Takt, PortC jeden Takt und PortD jeden achten Takt. Beenden Sie die Simulation nachdem jeder Port zehn mal geschrieben wurde.

Aufgabe 3.2.1 Scheduling

Implementieren Sie nun den obigen RAM aber benutzen Sie statt Round Robin Schedulingattribute.

Probieren Sie zunächst eine Lösung ohne Attribute.

Implementieren Sie abschließend eine möglichst schnell Implementierung mit Hilfe von Attributen. Ist Ihre Lösung auch schnell wenn sich die Datenfrequenz an den Ports ändert?