Einführung in Computer Microsystems Sommersemester 2015

Hörsaalübung 6: Klausurvorbereitung

TECHNISCHE UNIVERSITÄT DARMSTADT

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Hörsaalübung 6

Hörsaalübung 6

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fragen aus dem Forum
- Beispielaufgaben

Sprechstunden!

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sprechstunden mit Tutor in Raum E104
  - 16.7.2015 14:00 bis 16:30

Fragen: Wann welche Module?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Es gibt in Bluespec verschiedene Module mit „ähnlicher“ Funktion.
- `Wire, Reg, RWire, FIFO, ByPassFIFO, CReg` usw.
- Wann sollte welches eingesetzt werden?

Fragen: CReg

TECHNISCHE UNIVERSITÄT DARMSTADT

- Mehrere Änderungen an einem Register in einem Takt
- `mkCReg` erzeugt Array von `Reg` Interfaces
- Interfaces an höherer Position sehen Änderungen der vorherigen Stufen
- Erlaubt mehreren Rules ein Register zu bearbeiten

Fragen: Beispielmodul

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispielmodul: $r = 0 + a \times 2 - b$ falls $c > 6$ teile r durch c
- Neuberechnung wenn neue Parameter hinzugefügt werden
- Ergebnis muss im selben Takt zur Verfügung stehen

```
1    interface CRegTest;
2        method Action setParameter(Int#(32) a, Int#(32) b, Int#(32) c);
3
4        method Int#(32) getResult();
5    endinterface
```

Fragen: Typen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1   // Multiple changes to one register
2   Reg#(Int#(32)) theValue[5] <- mkCReg(5, 0);
3
4   // Wire with implicit condition and no type
5   PulseWire newCalculation <- mkPulseWire();
6
7   // Wires are only valid if there is a write in the same cycle
8   // Wire with Maybe as result
9   RWire#(Int#(32)) paramA <- mkRWireSBR();
10  // Wire with implicit valid check on Read
11  Wire#(Int#(32)) paramB <- mkWire();
12  Wire#(Int#(32)) paramC <- mkWire();
```

Fragen: Verschiedene Interfaces

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verschiedene Interfaces für verschieden Module
- Siehe BSV Referenz

```
1    method Action setParameter(Int#(32) a, Int#(32) b, Int#(32) c);
2        $display("(%0d) New parameters", $time);
3        newCalculation.send(); // PulseWire
4        paramA.wset(a);        // RWire
5        paramB <= b;           // Wire
6        paramC <= c;           // Wire
7    endmethod
```

Fragen: Berechnung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eine CReg Stufe pro Rule
- CReg Read sieht Zustand von vorheriger Regel (z.B. calc1 sieht Ergebnis von calc0)

```
1    rule calc0 (newCalculation);
2        theValue[0] <= 0;
3        $display("(%0d) CReg is now %d", $time, 0);
4    endrule
5
6    rule calc1 (paramA.wget() matches tagged Valid .v);
7        let newV = theValue[1] + (v * 2);
8        theValue[1] <= newV;
9        $display("(%0d) CReg is now %d", $time, newV);
10   endrule
```

Fragen: Berechnung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1    rule calc2;
2        let newV = theValue[2] - paramB;
3        theValue[2] <= newV;
4        $display("(%0d) CReg is now %d", $time, newV);
5    endrule
6
7    rule calc3 (paramC > 6);
8        let newV = theValue[3] / paramC;
9        theValue[3] <= newV;
10       $display("(%0d) CReg is now %d", $time, newV);
11   endrule
12
13   method Int#(32) getResult();
14       return theValue[4];
15   endmethod
```

Fragen: Testbench

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispielmodul: $r = 0 + a \times 2 - b$ falls $c > 6$ teile r durch c

```
1    (20) Result:            0
2    (30) Setting parameters a:          50 b:          20 c:          8
3    (30) New parameters
4    (30) CReg is now           0
5    (30) CReg is now         100
6    (30) CReg is now          80
7    (30) CReg is now          10
8    (30) Result in same cycle:          10
9    (40) Result next cycle cycle:          10
10   (50) Setting parameters a:          10 b:          30 c:          2
11   (50) New parameters
12   (50) CReg is now           0
13   (50) CReg is now          20
14   (50) CReg is now         -10
15   (50) Result in same cycle:         -10
16   (60) Result next cycle cycle:          -10
```

Fragen: Nested Interfaces

TECHNISCHE UNIVERSITÄT DARMSTADT

- Warum sind Nested Interfaces sinnvoll?
- Wiederverwendbarkeit!

```
1    interface SuperRAM;
2      interface Server#(Int#(32), Int#(32)) portA;
3      interface Server#(Int#(32), Int#(32)) portB;
4    endinterface
5    ....
6    SuperRAM ram <- mkSuperRAM();
7    Client#(Int#(32), Int#(32)) client0 <- mkSomeClient();
8    Client#(Int#(32), Int#(32)) client1 <- mkSomeClient();
9
10   mkConnection(ram.portA, client0);
11   mkConnection(ram.portB, client1);
```

Fragen: Nested Interfaces

TECHNISCHE UNIVERSITÄT DARMSTADT

- Alternative?

```
1   interface SuperRAM;
2     method Action putPortA(Int#(32));
3     method ActionValue#(Int#(32)) getPortA();
4     method Action putPortB(Int#(32));
5     method ActionValue#(Int#(32)) getPortB();
6   endinterface
7   ....
8   SuperRAM ram <- mkSuperRAM();
9   SomeClient client0 <- mkSomeClient();
10  SomeClient client1 <- mkSomeClient();
11
12  rule handlePortAPut;
13    let v <- client0.get();
14    ram.putPortA(v);
15  endrule
16  rule handlePortAGet;
17    let v <- ram.getPortA(v);
18    client0.put(v);
19  endrule
20  ...
```

Fragen: Praktische Verwendung von Automaten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Einsatz von StmtFSM außerhalb von Testbenches
- Vielfältiger Einsatz z.B. bei sequentiellem Prozessen
  - Warten bis fünf Minuten vergangen sind
  - Temperatur lesen
  - Sensorwert in ℃ umwandeln
  - Wert speichern
- TGDI: Moore und Mealy Automaten
- Synchronisation von parallelen Prozessen:
  - Daten Abrufen
  - Daten Verarbeiten
  - Daten Zurückschicken

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wann wird welches Attribut verwendet?
  - `mutually_exclusive`
  - `descending_urgency`
  - `preempts`

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- mutually_exclusive

```
1    interface MutuallyExclusive;
2        method Action putAB(UInt#(1) a, UInt#(1) b);
3    endinterface
4    module mkMutuallyExclusive(MutuallyExclusive);
5        Reg#(Bool) tickTock <- mkReg(False);
6        Reg#(UInt#(1)) counter <- mkReg(0);
7        Reg#(UInt#(1)) whenA <- mkReg(0);
8        Reg#(UInt#(1)) whenB <- mkReg(1);
9        // ... counter counts every cycle
10       rule tick (counter == whenA);
11           tickTock <= True;
12       endrule
13
14       rule tock (counter == whenB);
15           tickTock <= False;
16       endrule
17       // Interface sets whenA and whenB
18   endmodule
```

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- `mutually_exclusive`

```
1    Warning: "Test.bsv", line 93, column 8: (G0036)
2      Rule "tick" will appear to fire before "tock" when both fire in the same
3      clock cycle, affecting:
4        calls to tickTock.write vs. tickTock.write
5    Warning: "Test.bsv", line 93, column 8: (G0117)
6      Rule `tock' shadows the effects of `tick' when they execute in the same
7      clock cycle. Affected method calls:
8        tickTock.write
9      To silence this warning, use the `-no-warn-action-shadowing' flag.
```

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- mutually_exclusive

```
1    interface MutuallyExclusive;
2        method Action putAB(UInt#(1) a, UInt#(1) b);
3    endinterface
4    module mkMutuallyExclusive(MutuallyExclusive);
5        Reg#(Bool) tickTock <- mkReg(False);
6        Reg#(UInt#(1)) counter <- mkReg(0);
7        Reg#(UInt#(1)) whenA <- mkReg(0);
8        Reg#(UInt#(1)) whenB <- mkReg(1);
9        // ... counter counts every cycle
10       (* mutually_exclusive="tick, tock" *)
11       rule tick (counter == whenA);
12           tickTock <= True;
13       endrule
14
15       rule tock (counter == whenB);
16           tickTock <= False;
17       endrule
18       // Interface sets whenA and whenB
19   endmodule
```

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- `mutually_exclusive`

```
1    Error: "Test.bsv", line 105, column 28: (R0001)
2      Mutually exclusive rules (from the ME sets [RL_tick] and [RL_tock] )
         ↪   fired
3      in the same clock cycle.
```

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- descending_urgency

```
1   module mkDescendingUrgency(Empty);
2       Reg#(UInt#(32)) r <- mkReg(42);
3       Reg#(UInt#(32)) counter <- mkReg(0);
4       rule tick;
5           r <= r * 2;
6       endrule
7       rule tock ((counter & 1) == 0);
8           r <= r / 3;
9       endrule
10      rule count;
11          counter <= counter + 1;
12      endrule
13  endmodule
```

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- descending_urgency

```
1    Warning: "Test.bsv", line 147, column 8: (G0010)
2      Rule "tick" was treated as more urgent than "tock". Conflicts:
3        "tick" cannot fire before "tock": calls to r.write vs. r.read
4        "tock" cannot fire before "tick": calls to r.write vs. r.read
5    Warning: "Test.bsv", line 155, column 10: (G0021)
6      According to the generated schedule, rule `tock' can never fire.
```

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- descending_urgency

```
1    module mkDescendingUrgency(Empty);
2        Reg#(UInt#(32)) r <- mkReg(42);
3        Reg#(UInt#(32)) counter <- mkReg(0);
4        (* descending_urgency="tock,tick" *)
5        rule tick;
6            r <= r * 2;
7        endrule
8        rule tock ((counter & 1) == 0);
9            r <= r / 3;
10       endrule
11       rule count;
12           counter <= counter + 1;
13       endrule
14   endmodule
```

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- preempts

```
1    module mkPreempts(Empty);
2        Reg#(UInt#(32)) counter <- mkReg(0);
3
4        (* preempts="tock,tick" *)
5        rule tick; // implicit condition (!tock_fires)
6            $display("(%0d) Tick", $time);
7        endrule
8
9        rule tock ((counter & 1) == 0);
10           $display("(%0d) Tock", $time);
11       endrule
12
13       rule count;
14           counter <= counter + 1;
15       endrule
16   endmodule
```

Fragen: Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

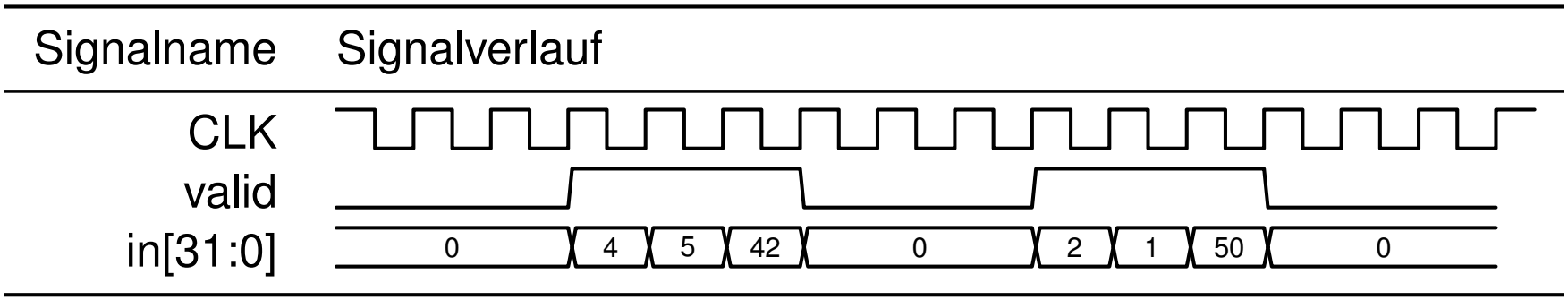
- `preempts`

```
1     (10) Tock
2     (20) Tick
3     (30) Tock
4     (40) Tick
5     (50) Tock
6     (60) Tick
7     (70) Tock
8     (80) Tick
9     (90) Tock
10    (100) Tick
```

Beispielaufgabe: Bluespec FIFO

TECHNISCHE UNIVERSITÄT DARMSTADT

Gegeben sei folgendes Input-Pattern.

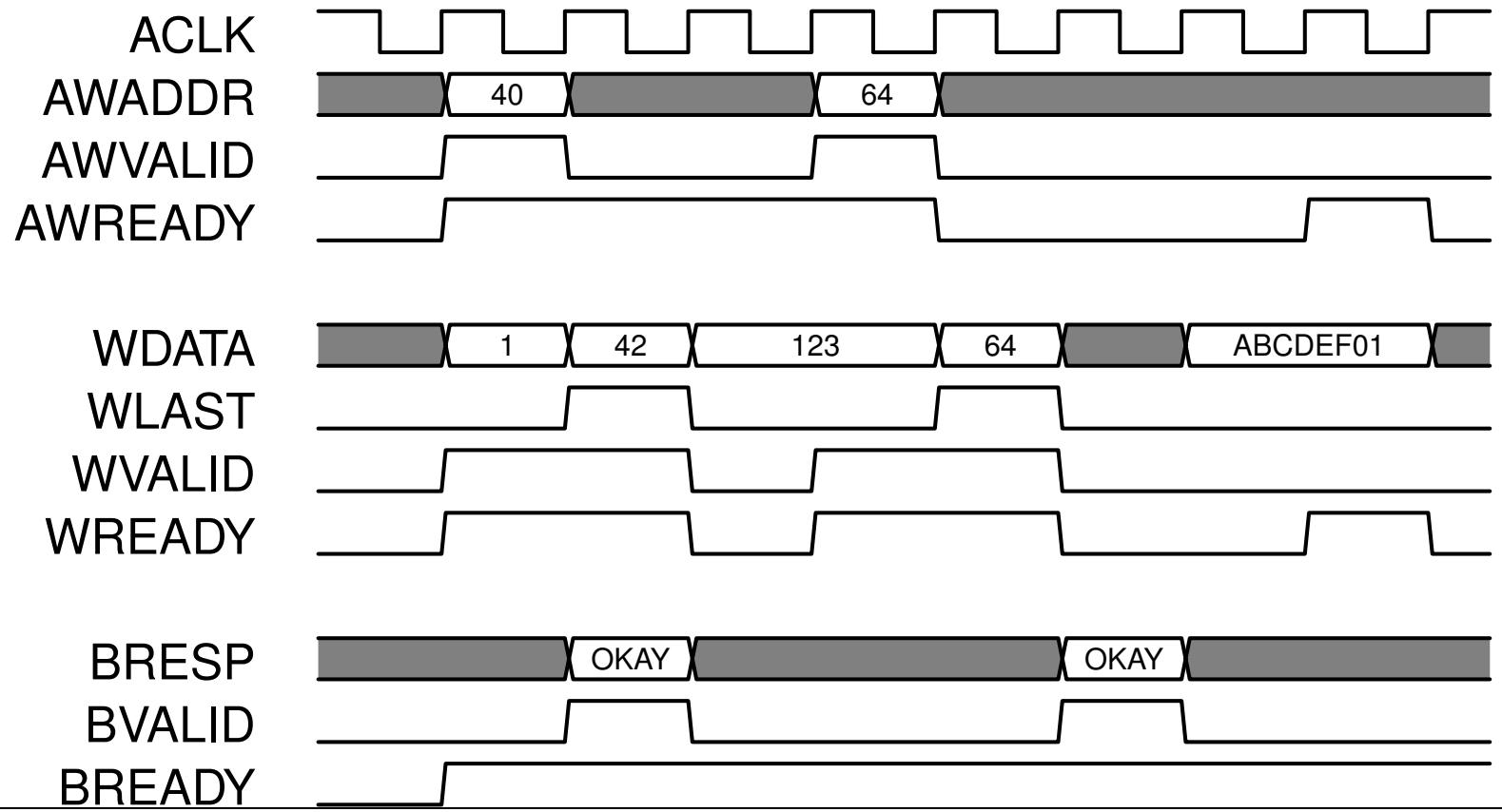


Dieses Pattern setzt sich auf die selber Art weiter fort. Jedes Datum liegt immer für genau einen Takt an und es werden immer drei Daten hintereinander übertragen. Die Berechnung eines Datums benötigt zwei Takte. Neue Daten kommen frühestens drei Takte nach dem letzten Datum eines Bursts.
Instantiieren Sie eine FIFO in Bluespec die die Eingabedaten möglichst Speichereffizient aufnehmen kann.

Beispielaufgabe: AXI Übertragung

TECHNISCHE UNIVERSITÄT DARMSTADT

Welche Daten werden bei diesem (vereinfacht dargestellten) AXI Transfer an welche Speicherstelle übertragen (AWLEN=1)?



Beispielaufgabe: AXI Übertragung

TECHNISCHE UNIVERSITÄT DARMSTADT

Richtung:

Transfer	Adresse	Daten

Beispielaufgabe: AXI Übertragung

TECHNISCHE UNIVERSITÄT DARMSTADT

Richtung: Master schreibt Slave

Transfer	Adresse	Daten
0	40	1
1	44	42
2	64	123
3	68	64

Beispielaufgabe: AXI Theorie

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Nennen Sie eine Eigenschaft bezüglich der Kommunikationsrichtungen von AXI.

Beispielaufgabe: AXI Theorie

TECHNISCHE UNIVERSITÄT DARMSTADT

Nennen Sie eine Eigenschaft bezüglich der Kommunikationsrichtungen von AXI.

- Getrennte Lese- und Schreibrichtungen
- Müssen nicht immer vorhanden sein

Beispielaufgabe: System on Chip

TECHNISCHE UNIVERSITÄT DARMSTADT

Nennen Sie zwei Vorteile und zwei Nachteile die ein rekonfigurierbares System-on-Chip bietet.
Vorteile:

Nachteile:

Beispielaufgabe: System on Chip

TECHNISCHE UNIVERSITÄT DARMSTADT

Nennen Sie zwei Vorteile und zwei Nachteile die ein rekonfigurierbares System-on-Chip bietet.
Vorteile:

- Ein Chip für viele Anwendungen
- Anpassbar auf künfigte Entwicklungen

Nachteile:

- Niedriger Takt
- Größerer Platzbedarf (im Vergleich mit fest verdrahteter Funktion)

Weitere Themen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Direct-Mapped-Cache (Addresse in Tag, Slot, Byte Offset)
- Cache in Mehrprozessorsystemen
- System on Chip
- Zynq
- Bluespec Konflikte auflösen