Technische Universität Darmstadt
FG Eingebettete Systeme und
ihre Anwendungen (ESA)

Prof. Dr. Andreas Koch
Holger Lange

18.06.2007

Einführung in Computer Microsystems
5. Aufgabenblatt
Sommersemester 2007

Aufgabe 1: Verkaufsautomat „Multicoin"

Der aus der Vorlesung (Foliensatz 4 ab Folie 82) bekannte Eis-Verkaufsautomat soll zusätzlich 10- und 20-Cent Münzen sowie 2-Euro Münzen akzeptieren. Dazu erhält er ein auf drei Bit verbreitertes **COIN**-Signal, welches wie bisher symbolisch dekodiert werden soll. Das Timing der Münzeingabe sowie die übrigen Spezifikationen bleiben gleich, die Eisausgabe darf einen Takt später als bisher erfolgen. Implementieren Sie das Verilog-Modell des erweiterten Mealy-Zustandsautomaten mit zugehöriger Testbench, welche zusätzlich die neuen Münzeingaben testen soll. Die Testausgabe erfolgt wie bisher in der bekannten Tabellenform. Achten Sie darauf, dass bei der Sythese keine Latches enstehen.

Tipp: Bei der nun vorhandenen Zahl von Kombinationsmöglichkeiten des Münzeinwurfs ist es nicht sinnvoll, den eingeworfenen Geldbetrag als symbolische Zustände zu kodieren. Verwenden Sie stattdessen ein internes Summenregister.

Lösung:

Mealy-Verkaufsautomat in der Multicoin-Version:

```
// Verkaufsautomat
`timescale 1 ns / 1 ps
module iglu_multicoin (
  input wire       CLOCK, // Takt
                   RESET, // Reset
  input wire [2:0] COIN,  // eingeworfene Muenze
  output reg       ICE);  // Warenausgabe

  `include "iglu_multicoin_defs.v"
```

```
  // interne Variable
  reg       PRESENT,      // jetziger Zustand
            NEXT;         // naechster Zustand

  reg [5:0] SUM;          // eingezahlter Geldbetrag
                          // Einheit 10 Cent
  reg [5:0] NEXT_SUM;     // neue Summe
                          // Einheit 10 Cent

  always @(PRESENT, COIN, SUM)      // <-- flankenfrei!
    case (PRESENT)
      INSERT_COINS: begin           // Muenzeinwurf
        case (COIN)
          X0:                       // kein Einwurf
            NEXT_SUM = SUM + 0;
          X10:                      // 10 Cent
            NEXT_SUM = SUM + 1;
          X20:                      // 20 Cent
            NEXT_SUM = SUM + 2;
          X50:                      // 50 Cent
            NEXT_SUM = SUM + 5;
          X100:                     // 1 Euro
            NEXT_SUM = SUM + 10;
          X200:                     // 2 Euro
            NEXT_SUM = SUM + 20;
          default:                  // Don't care, Latch vermeiden
            NEXT_SUM = 'bx;
        endcase
        if (SUM >= Price)           // Ueberzahlung wird ignoriert
          {NEXT,ICE} = {ICECREAM, No};
        else
          {NEXT,ICE} = {INSERT_COINS, No};
      end
      ICECREAM: begin               // genug bezahlt
        NEXT_SUM = 0;               // eingezahlter Betrag wieder auf 0
        {NEXT,ICE} =                // Warenausgabe
          {INSERT_COINS, Yes};
      end
    endcase

  // naechsten Zustand mit jedem Takt endgueltig zum jetzigen machen;
  // synchrones Reset
  //
  always @(posedge CLOCK)             // <-- flankengesteuert!
    if (RESET == 1'b1) begin
      PRESENT <= INSERT_COINS;        // Anfangszustand
      SUM <= 0;
    end
    else begin
      PRESENT <= NEXT;
      SUM <= NEXT_SUM;
    end

endmodule // iglu_multicoin
```

Testbench:

```
`timescale 1 ns / 1 ps
module tb_iglu_multicoin;
  reg CLOCK, RESET;
  reg [2:0] COIN;
  wire ICE;

  `include "iglu_multicoin_defs.v"

  // Instanz des Verkaufsautomaten
  iglu_multicoin UUT (.CLOCK(CLOCK), .RESET(RESET),
                      .COIN(COIN), .ICE(ICE));

  // Takt
  always
    #20 CLOCK = ~CLOCK;

  // Ergebnis drucken
  initial begin
    $display ("␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Zeit␣Reset␣Eisausgabe\n");
    $monitor ("%d␣␣␣%d␣␣␣␣␣%d", $time, RESET, ICE);
  end

  // Stimuli anlegen: Muenzen eingeben
  initial begin
    CLOCK = 0; COIN = X0; RESET = 1'b1;
    #50 RESET = 1'b0;
    @(negedge CLOCK);
    // drei Fuenfziger
    #80 $display ("Einwurf␣50␣Ct"); COIN = X50; #40 COIN = X0;
    #80 $display ("Einwurf␣50␣Ct"); COIN = X50; #40 COIN = X0;
    #80 $display ("Einwurf␣50␣Ct"); COIN = X50; #40 COIN = X0;
    // einen Fuenfziger, dann einen Euro
    #160 $display ("Einwurf␣50␣Ct"); COIN = X50; #40 COIN = X0;
    #80 $display ("Einwurf␣100␣Ct"); COIN = X100; #40 COIN = X0;
    // zwei Euro (keine Rueckgabe!)
    #160 $display ("Einwurf␣100␣Ct"); COIN = X100; #40 COIN = X0;
    #80 $display ("Einwurf␣100␣Ct"); COIN = X100; #40 COIN = X0;
    // einen Euro, dann einen Fuenfziger
    #160 $display ("Einwurf␣100␣Ct"); COIN = X100; #40 COIN = X0;
    #80 $display ("Einwurf␣50␣Ct"); COIN = X50; #40 COIN = X0;
    // zwei Euro
    #160 $display ("Einwurf␣200␣Ct"); COIN = X200; #40 COIN = X0;
    // 6 Zehner, 2 Zwanziger, ein Fuenziger
    #160 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
    #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
    #80 $display ("Einwurf␣20␣Ct"); COIN = X20; #40 COIN = X0;
    #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
    #80 $display ("Einwurf␣20␣Ct"); COIN = X20; #40 COIN = X0;
    #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
    #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
    #80 $display ("Einwurf␣50␣Ct"); COIN = X50; #40 COIN = X0;
    #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
    #80 $stop;
  end
```

```
endmodule
```

Testausgabe:

```
                Zeit Reset Eisausgabe
                   0   1     x
                  20   1     0
                  50   0     0
Einwurf 50 Ct
Einwurf 50 Ct
Einwurf 50 Ct
                 460   0     1
                 500   0     0
Einwurf 50 Ct
Einwurf 100 Ct
                 780   0     1
                 820   0     0
Einwurf 100 Ct
Einwurf 100 Ct
                1100   0     1
                1140   0     0
Einwurf 100 Ct
Einwurf 50 Ct
                1420   0     1
                1460   0     0
Einwurf 200 Ct
                1620   0     1
                1660   0     0
Einwurf 10 Ct
Einwurf 10 Ct
Einwurf 20 Ct
Einwurf 10 Ct
Einwurf 20 Ct
Einwurf 10 Ct
Einwurf 10 Ct
Einwurf 50 Ct
Einwurf 10 Ct
                2780   0     1
                2820   0     0
```

Aufgabe 2: Wechselgeld

Aufgrund der Unzufriedenheit vieler Kunden mit der fehlenden Wechselgeldausgabe ist der Umsatz des Eis-Verkaufsautomaten aus Aufgabe 1 eingebrochen. Zu allem Überfluss verkauft ein Eisstand gegenüber das Konkurrenzprodukt für 1,40 €. Als Chefdesigner des Eisautomaten erhalten Sie den Auftrag, folgende Maßnahmen aufbauend auf dem „Multicoin"-Automaten umzusetzen:

a) Wechselgeldausgabe

Ein zusätzliches Ausgangsignal `OUTCOIN` zeigt in derselben symbolischen Kodierung wie `COIN` an, dass eine Münze des entsprechenden Wertes an den Kunden ausgegeben werden

soll. Das Timing dieses Signals soll identisch wie bei **COIN** sein, um die Münzausgabeeinheit korrekt anzusteuern. Achten Sie auf die Pausen von einem Takt zwischen den eigentlichen Münzwerten (**X10**, **X20**, ...), während denen **OUTCOIN** den Wert **X0** annehmen soll. Gehen Sie davon aus, dass immer eine unbegrenzte Menge an Münzen jeden Wertes als Wechselgeld vorhanden ist. Tipp: Wegen der 1-2-5 Münzstückelung bietet sich ein Greedy-Algorithmus für die Bestimmung der Wechselmünzen an.

b) Abbruch des Kaufs durch den Benutzer

Eine „1" auf dem zusätzlichen Eingang **CANCEL** zeigt an, dass der Kunde nun doch kein Eis will und sein eventuell schon eingeworfenes Geld wiederhaben möchte. Diese Funktion ist nur solange möglich, bis mindestens der Kaufpreis für ein Eis eingeworfen wurde. In diesem Fall wird das Eis sofort ausgegeben und das Geld abzüglich des Wechselgeldes einbehalten, das Wechselgeld anschließend ausgegeben. Nutzen Sie die Wechselgeldrückgabe aus a) für die Implementierung der Abbruchfunktion.

c) Preissenkung des Eises auf 1,30 €

Kodieren Sie hierzu den Preis für ein Eis als Parameter, um ihn später leichter anpassen zu können.

Implementieren Sie das Verilog-Modell des Mealy-Zustandsautomaten aufbauend auf Aufgabe 1. Die Mealy-Eigenschaft muss trotz der erweiterten Funktionalität erhalten bleiben. Erweitern Sie die Testbench um den Test der Wechselgeld- und Abbruchfunktionen. Die tabellarische Testausgabe soll zusätzlich das Signal **OUTCOIN** enthalten. Passen Sie die Wartezeiten nach einer Münzeinwurfsfolge den bedingt durch die Geldrückgabe verlängerten Reaktionszeiten des Automaten an. Achten Sie auch bei dieser Aufgabe darauf, dass bei der Synthese keine Latches entstehen.

Lösung:

Mealy-Verkaufsautomat „Multicoin-Change" mit Wechselgeld- und Abbruchfunktion:

```
// Verkaufsautomat mit Wechselgeldrueckgabe und Abbruchfunktion
`timescale 1 ns / 1 ps
module iglu_multicoin_change (
  input wire       CLOCK,    // Takt
                   RESET,    // Reset
                   CANCEL,   // Abbruch mit Geldrueckgabe
  input wire [2:0] COIN,     // eingeworfene Muenze
  output reg       ICE,      // Warenausgabe
  output reg [2:0] OUTCOIN); // herauszugebene Muenze

  `include "iglu_multicoin_change_defs.v"

  // interne Variable
  reg [1:0] PRESENT,      // jetziger Zustand
            NEXT;         // naechster Zustand

  reg [5:0] SUM;          // eingezahlter Geldbetrag
                          // Einheit 10 Cent
  reg [5:0] NEXT_SUM;     // neue Summe
                          // Einheit 10 Cent
```

```
always @(PRESENT, COIN, SUM, CANCEL)// <-- flankenfrei!
  case (PRESENT)
    INSERT_COINS: begin              // Muenzeinwurf
      case (COIN)
        X0:                          // kein Einwurf
          NEXT_SUM = SUM + 0;
        X10:                         // 10 Cent
          NEXT_SUM = SUM + 1;
        X20:                         // 20 Cent
          NEXT_SUM = SUM + 2;
        X50:                         // 50 Cent
          NEXT_SUM = SUM + 5;
        X100:                        // 1 Euro
          NEXT_SUM = SUM + 10;
        X200:                        // 2 Euro
          NEXT_SUM = SUM + 20;
        default:                     // Don't care, Latch vermeiden
          NEXT_SUM = 'bx;
      endcase
      if (SUM >= Price)              // Eis ausgeben mit Wechselgeld
        {NEXT,ICE,OUTCOIN} = {ICECREAM,No,X0};
      else if (CANCEL)               // Abbruch durch Kunden, Geldrückgabe
        {NEXT,ICE,OUTCOIN} = {CHANGE,No,X0};
      else
        {NEXT,ICE,OUTCOIN} = {INSERT_COINS,No,X0};
    end
    ICECREAM: begin                  // genug bezahlt
      NEXT_SUM = SUM - Price;        // Restgeldbetrag
      {NEXT,ICE,OUTCOIN} =           // Warenausgabe, Wechselgeld
        {CHANGE,Yes,X0};
    end
    CHANGE: begin                    // Wechselgeld ausgeben, Greedy
      if (SUM >= 10) begin           // ein Euro ausgeben
        NEXT_SUM = SUM - 10;
        {NEXT,ICE,OUTCOIN} = {WAIT,No,X100};
      end
      else if (SUM >= 5) begin       // 50 Cent
        NEXT_SUM = SUM - 5;
        {NEXT,ICE,OUTCOIN} = {WAIT,No,X50};
      end
      else if (SUM >= 2) begin       // 20 Cent
        NEXT_SUM = SUM - 2;
        {NEXT,ICE,OUTCOIN} = {WAIT,No,X20};
      end
      else if (SUM == 1) begin       // 10 Cent
        NEXT_SUM = SUM - 1;
        {NEXT,ICE,OUTCOIN} = {WAIT,No,X10};
      end
      else begin                     // Alles Wechselgeld ausgegeben
        NEXT_SUM = SUM;
        {NEXT,ICE,OUTCOIN} = {INSERT_COINS,No,X0};
      end
    end
    WAIT: begin                      // Pause zwischen Muenzausgaben
      NEXT_SUM = SUM;
```

```
        {NEXT,ICE,OUTCOIN} = {CHANGE,No,X0};
      end
    endcase

  // naechsten Zustand mit jedem Takt endgueltig zum jetzigen machen;
  // synchrones Reset
  //
  always @(posedge CLOCK)              // <-- flankengesteuert!
    if (RESET == 1'b1) begin
      PRESENT <= INSERT_COINS;         // Anfangszustand
      SUM <= 0;
    end
    else begin
      PRESENT <= NEXT;
      SUM <= NEXT_SUM;
    end

endmodule // iglu_multicoin_change
```

Testbench:

```
`timescale 1 ns / 1 ps
module tb_iglu_multicoin_change;
  reg CLOCK, RESET, CANCEL;
  reg [2:0] COIN;
  wire ICE;
  wire [2:0] OUTCOIN;

  `include "iglu_multicoin_change_defs.v"

  // Instanz des Verkaufsautomaten
  iglu_multicoin_change UUT
    (.CLOCK(CLOCK), .RESET(RESET), .CANCEL(CANCEL),
     .COIN(COIN), .ICE(ICE), .OUTCOIN(OUTCOIN));

  // Takt
  always
    #20 CLOCK = ~CLOCK;

  // Ergebnis drucken
  initial begin
    $display ("                Zeit  Reset  Eisausgabe  Muenzausgabe\n");
    $monitor ("%d   %d       %d           %d", $time, RESET, ICE, OUTCOIN);
  end

  // Stimuli anlegen: Muenzen eingeben
  initial begin
    CLOCK = 0; COIN = X0; CANCEL = 1'b0; RESET = 1'b1;
    #50 RESET = 1'b0;
    @(negedge CLOCK);
    // drei Fuenfziger
    #80 $display ("Einwurf 50 Ct"); COIN = X50; #40 COIN = X0;
    #80 $display ("Einwurf 50 Ct"); COIN = X50; #40 COIN = X0;
    #80 $display ("Einwurf 50 Ct"); COIN = X50; #40 COIN = X0;
    // einen Fuenfziger, dann einen Euro
    #240 $display ("Einwurf 50 Ct"); COIN = X50; #40 COIN = X0;
```

```
        #80 $display ("Einwurf␣100␣Ct"); COIN = X100; #40 COIN = X0;
        // zweimal ein Euro
        #240 $display ("Einwurf␣100␣Ct"); COIN = X100; #40 COIN = X0;
        #80 $display ("Einwurf␣100␣Ct"); COIN = X100; #40 COIN = X0;
        // einen Euro, dann einen Fuenfziger
        #320 $display ("Einwurf␣100␣Ct"); COIN = X100; #40 COIN = X0;
        #80 $display ("Einwurf␣50␣Ct"); COIN = X50; #40 COIN = X0;
        // zwei Euro
        #240 $display ("Einwurf␣200␣Ct"); COIN = X200; #40 COIN = X0;
        // 1 Euro, 1 Zwanziger, Abbruch
        #280 $display ("Einwurf␣100␣Ct"); COIN = X100; #40 COIN = X0;
        #80 $display ("Einwurf␣20␣Ct"); COIN = X20; #40 COIN = X0;
        #80 $display ("Abbruch"); CANCEL = 1'b1; #40 CANCEL = 1'b0;
        // einen Fuenziger, drei Zehner, Abbruch
        #320 $display ("Einwurf␣50␣Ct"); COIN = X50; #40 COIN = X0;
        #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
        #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
        #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
        #80 $display ("Abbruch"); CANCEL = 1'b1; #40 CANCEL = 1'b0;
        // 4 Zehner, 2 Zwanziger, ein Fuenziger (keine Rueckgabe!)
        #400 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
        #80 $display ("Einwurf␣20␣Ct"); COIN = X20; #40 COIN = X0;
        #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
        #80 $display ("Einwurf␣20␣Ct"); COIN = X20; #40 COIN = X0;
        #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
        #80 $display ("Einwurf␣50␣Ct"); COIN = X50; #40 COIN = X0;
        #80 $display ("Einwurf␣10␣Ct"); COIN = X10; #40 COIN = X0;
        #160 $stop;
      end

    endmodule
```

Testausgabe:

```
                Zeit Reset Eisausgabe Muenzausgabe
                   0   1       x           x
                  20   1       0           0
                  50   0       0           0
Einwurf 50 Ct
Einwurf 50 Ct
Einwurf 50 Ct
                 460   0       1           0
                 500   0       0           2
                 540   0       0           0
Einwurf 50 Ct
Einwurf 100 Ct
                 860   0       1           0
                 900   0       0           2
                 940   0       0           0
Einwurf 100 Ct
Einwurf 100 Ct
                1260   0       1           0
                1300   0       0           3
                1340   0       0           0
                1380   0       0           2
```

```
                1420    0       0           0
Einwurf 100 Ct
Einwurf 50 Ct
                1740    0       1           0
                1780    0       0           2
                1820    0       0           0
Einwurf 200 Ct
                2020    0       1           0
                2060    0       0           3
                2100    0       0           0
                2140    0       0           2
                2180    0       0           0
Einwurf 100 Ct
Einwurf 20 Ct
Abbruch
                2540    0       0           4
                2580    0       0           0
                2620    0       0           2
                2660    0       0           0
Einwurf 50 Ct
Einwurf 10 Ct
Einwurf 10 Ct
Einwurf 10 Ct
Abbruch
                3380    0       0           3
                3420    0       0           0
                3460    0       0           2
                3500    0       0           0
                3540    0       0           1
                3580    0       0           0
Einwurf 10 Ct
Einwurf 20 Ct
Einwurf 10 Ct
Einwurf 20 Ct
Einwurf 10 Ct
Einwurf 50 Ct
Einwurf 10 Ct
                4580    0       1           0
                4620    0       0           0
```