Übung zur Vorlesung Einführung in Computer Microsystems

Prof. Dr. A. Koch
Thorsten Wink

TECHNISCHE UNIVERSITÄT DARMSTADT

Sommersemester 11
Übungsblatt 3

Aufgabe 3.1 Logik, Latch, Register

Geben Sie für alle folgenden `reg`-Variablen an, ob sie bei der Synthese in Latches, Flip-Flops oder kombinatorische Logik übersetzt werden. Begründen Sie Ihre Antworten mit den Kriterien für potenzielle Register.

Aufgabe 3.1.1 a)

```
module a (input CLOCK, I, output reg O);
  reg T;
  always @(CLOCK, I)
    if (CLOCK) begin
      T = I;
      O = T;
    end
endmodule
```

Aufgabe 3.1.2 b)

```
module b (
  input wire       A,
  input wire [2:0] I,
  output reg [7:0] OCTAL);

always @(posedge A)
  case (I)
    3'h0: OCTAL = 8'b00000001;
    3'h1: OCTAL = 8'b00000010;
    3'h2: OCTAL = 8'b00000100;
    3'h3: OCTAL = 8'b00001000;
    3'h4: OCTAL = 8'b00010000;
    3'h5: OCTAL = 8'b00100000;
    3'h6: OCTAL = 8'b01000000;
    3'h7: OCTAL = 8'b10000000;
  endcase
endmodule
```

Aufgabe 3.1.3 c)

```
module c (
  input wire [3:0] I,
  output reg [7:0] OCTAL);
```

```
always @(I)
  case (I)
    4'h0: OCTAL = 8'b00000001;
    4'h1: OCTAL = 8'b00000010;

    4'h2: OCTAL = 8'b00000100;
    4'h3: OCTAL = 8'b00001000;
    4'h4: OCTAL = 8'b00010000;
    4'h5: OCTAL = 8'b00100000;
    4'h6: OCTAL = 8'b01000000;
    4'h7: OCTAL = 8'b10000000;
  endcase
endmodule
```

Aufgabe 3.2 BCD nach Binär Konverter

Implementieren Sie den folgenden Pseudo-Code für einen BCD nach Binär Konverter als Verhaltensbeschreibung in Verilog HDL. Die Länge der BCD-Zahl sei 24 Bit. Achten Sie darauf, dass bei der Synthese keine Latches entstehen. Testen Sie die Funktion ihres Moduls mit ISE.

Pseudo-Code für BCD nach Binär Konverter:

a) `bcd` ← `bcd_data_input`

b) `bin` ← 0 (gleiche Bitbreite wie `bcd`)

c) Für `count` ← 1 bis Bitbreite von `bcd` iteriere:

   3.1. `{bcd, bin}` ← `{bcd, bin}` >> 1

   3.2. Für jede 4-Bit Folge (3...0, 7...4, ...) in `bcd` iteriere
   Wenn die 4-Bit Folge größer 7, dann ziehe 3 von dieser Folge ab

d) `bin` enthält die konvertierte Zahl

Aufgabe 3.3 Verkaufsautomat Multicoin

Der aus der Vorlesung (Foliensatz 4 ab Folie 86) 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.

Aufgabe 3.4 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 3. 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.

Diese Hausaufgaben müssen bis 3.6.11, 18:00 über das Moodle-System abgegeben werden.

Hausaufgabe 3.1 Modulare Multiplikation (10 Punkte)

Bei vielen Kryptographieverfahren wird eine besondere Multiplikation benötigt: Die modulare Multiplikation. Hierbei wird das Ergebnis der Multiplikation modulo des Körperpolynoms gerechnet, damit das Ergebnis auch wieder im Körper liegt.
Wir verwenden für die Koeffizienten der Polynome den Körper $\mathbb{GF}(2)$. Dieser besteht aus den Elementen 0 und 1. Es gelten die folgenden Regeln:

a	b	+	·
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Die Polynome selbst sind Elemente des Körpers $\mathbb{GF}(2^{11})$. Dies sind Polynome vom Grad 10 mit Koeffizienten aus dem $\mathbb{GF}(2)$. Diese Polynome können als einfacher Bitstring dargestellt werden. Das Polynom $x^{10}+x^5+x^3+1$ entspricht etwa dem String 10000101001. Die Multiplikation zweier Polynome folgt den Regeln der normalen Polynommultiplikation mit anschließender Reduktion modulo des Körperpolynoms.
Beispiel:
$(x^{10}+x^5+x^3+1)\star(x^3+1)$
Darstellung als Bitstring:
$10000101001\star 00000001001 \mod 100000000101$
$= 10010101100001 \mod 100000000101$
$= 10101110101$

Hausaufgabe 3.1.1 a)

Erstellen Sie ein Modul in Verilog, welches 2 Polynome aus dem Körper $\mathbb{GF}(2^{11})$ vom Grad 10 multipliziert und das Ergebnis modulo des Körperpolynoms vom Grad 11 ausgibt. Das Modul soll folgende Schnittstelle haben:

```
module modmul(
  input  wire         clk,       //Takt
  input  wire         reset,     //Reset
  input  wire [11:0]  polynom,   //Körperpolynom vom Grad 11
  input  wire [10:0]  datain_a,  //Eingang A
  input  wire [10:0]  datain_b,  //Eingang B
  input  wire         enable,    //= 1, wenn gültige Daten am Eingang liegen
  output wire [10:0]  dataout,   //Ergebnis
  output wire         valid      //= 1, falls dataout gültig
  )
```

Der Multiplikationsoperator $\star$ darf nicht verwendet werden.

Hausaufgabe 3.1.2 b)

Beschreiben Sie Ihre Lösungsidee in Worten (extra pdf-Datei). Gehen Sie auch auf mögliche Schwächen Ihrer gewählten Lösung ein. Achten Sie auf eine effiziente Implementierung. Das Modul muss synthetisierbar sein. Überprüfen Sie die korrekte Funktion durch eine Testbench.

Hausaufgabe 3.1.3 c)

In einer weiteren Variante soll das Körperpolynom fest vorgegeben sein. Der Eingang `polynom` entfällt. Als Körperpolynom soll das Polynom $x^{11} + x^2 + 1$ verwendet werden. Erstellen Sie ein Modul `modmul_fix`. Achten Sie auf eine möglichst geringe Latenz.

Plagiarismus

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Weitere Infos unter www.informatik.tu-darmstadt.de/plagiarism