Übung zur Vorlesung Einführung in Computer Microsystems

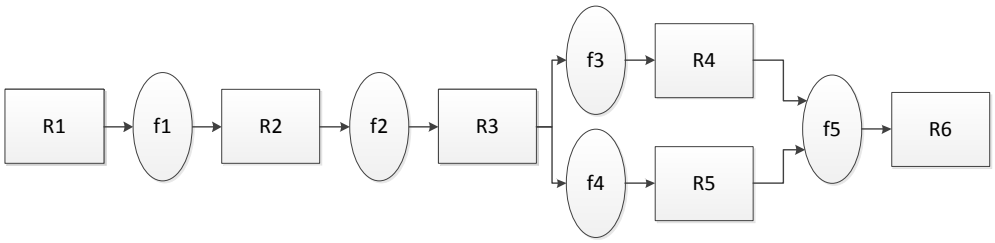
Prof. Dr. A. Koch
Thorsten Wink

TECHNISCHE UNIVERSITÄT DARMSTADT

Sommersemester 11
Übungsblatt 2

Aufgabe 2.1 Register-Transfer-Logik

Gegeben ist folgende Pipeline in Register-Transfer-Logik:



Die kombinatorische Logik zwischen den Registern soll folgende Funktionen realisieren.

- f_1: verdoppeln
- f_2: plus 5
- f_3: quadrieren
- f_4: minus 3
- f_5: Wenn R4 gerade, dann leite R4 weiter, ansonsten R5

Aufgabe 2.1.1 Funktionen

Beschreiben Sie die Pipeline in Verilog HDL. Die Funktionen sollen in Verilog HDL als `function` realisiert werden.
Die Register R1, R2, R3, R4, R5 und R6 sind jeweils 8-Bit breit. Überlaufen der Register kann vernachlässigt werden. Das Register R1 soll mit einem Wert geladen werden können. Wenn ein Steuersignal `ldreg1` gesetzt ist, soll über den Port `in` von auSSen ein Wert in das Register R1 geladen werden. Das Ergebnis der Berechnung soll über den Port `out` nach aussen geführt werden. Weisen Sie die korrekte Funktionsweise der Pipeline durch Simulation nach.

Aufgabe 2.1.2 asynchroner Reset

Die Pipeline aus Aufgabenteil a) soll um einen asynchronen Reset-Eingang `areset` erweitert werden, mit dem die Pipeline zurückgesetzt werden kann. Wird `areset` gesetzt, soll die Berechnung sofort gestoppt werden und solange keine neue Berechnung begonnen werden, bis `areset` wieder den Wert 0 annimmt.

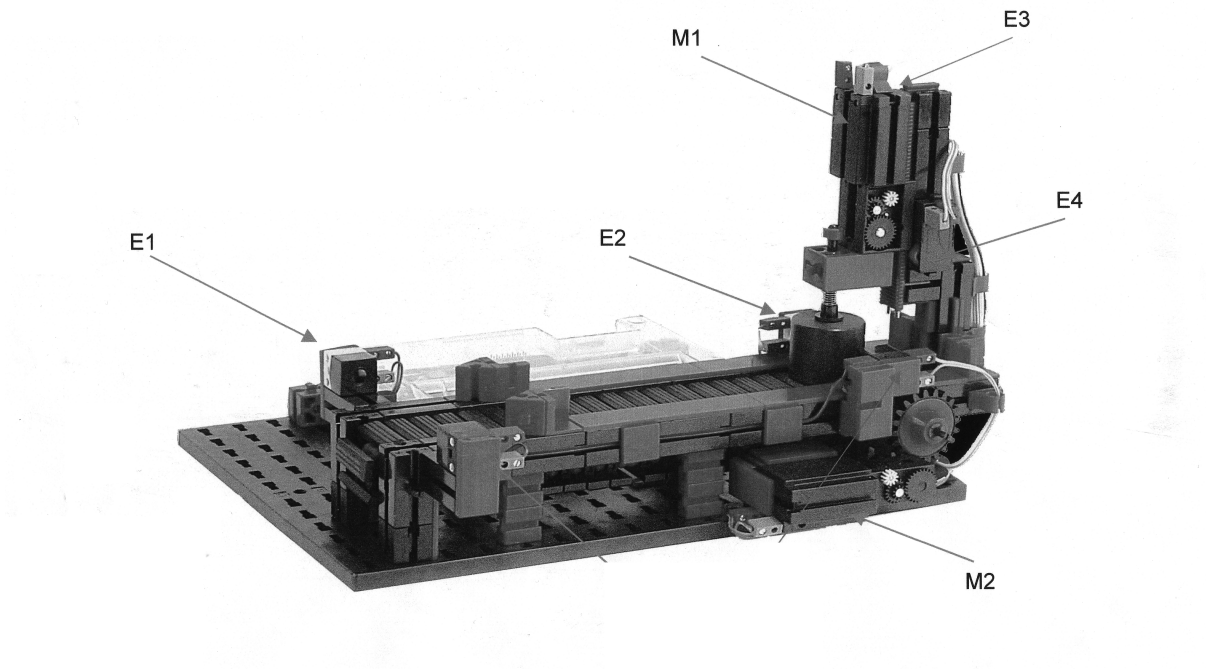
Aufgabe 2.1.3 Valid

Da die Pipeline einige Takte benötigt um das Ergebnis zu berechnen, liegen am Ausgang zwischenzeitlich falsche Werte an. Das Module aus Aufgabenteil b) soll deshalb um ein Signal `valid` erweitert werden. `valid` soll genau dann den Wert 1 haben, wenn der Wert an `out` ein korrektes Ergebnis eines geladenen Wertes ist.

Übung zur Vorlesung Einführung in Computer Microsystems

Aufgabe 2.2 Entwurf der Steuerung einer Stanzmaschine

Die folgende Abbildung zeigt eine Stanzmaschine.



Die zu entwerfende Steuerung soll folgende Funktion realisieren.
Wird das zu stanzende Werkstück, z. B. von einem Gabelstapler auf das Förderband zwischen die Lichtschranke (E1) gelegt, so wird das Förderband (Motor M2) gestartet. Das Förderband läuft solange bis die hintere Lichtschranke (E2) erreicht ist. Danach kann der Stanzvorgang (Motor M1) gestartet werden. Der Ende-Schalter (E4) signalisiert, dass das Stanzen erfolgt ist. Danach muss die Stanze wieder in die Ausgangsposition gefahren werden. Das Erreichen signalisiert ein Ende-Schalter (E3). SchlieSSlich soll das Werkstück wieder zurück transportiert werden. Es kann davon ausgegangen werden, dass sich immer nur ein Werkstück auf dem Förderband befindet. Die Steuerung soll, um Unfälle zu vermeiden, beim Start eines Stanzvorgangs kontrollieren, ob sich der Stempel in seiner Grundposition befindet (E3).

Aufgabe 2.2.1 Zustandsgraph

Entwerfen Sie für die Stanzmaschine den Zustandsgraphen des Automaten (Moore) mit symbolischen Eingaben, Ausgaben und Zuständen. Zeichnen Sie den Zustandsgraphen. Kodieren Sie die Zustände des Steuerwerkes.

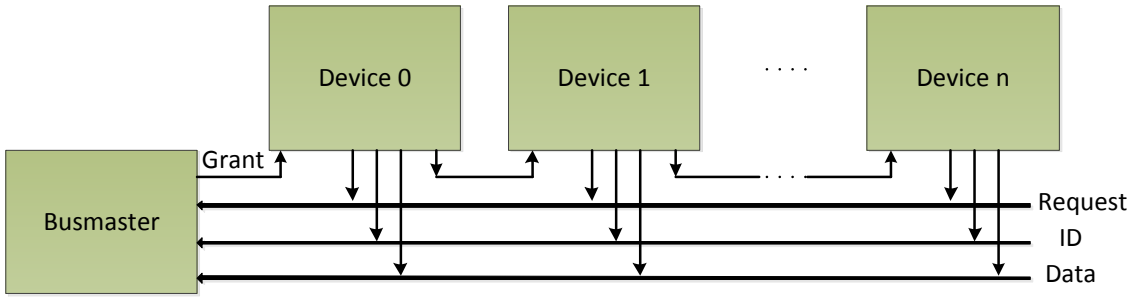
Aufgabe 2.2.2 Implementierung

Implementieren Sie den Moore-Automaten in Verilog HDL. Weisen Sie die korrekte Funktionsweise des Automaten durch Simulation nach. Wieviele Flip-Flops werden zur Realisierung Ihres Schaltwerks benötigt? Die Kodierung der Zustände ist Ihnen überlassen.

Diese Hausaufgaben müssen bis 20.5.11, 18:00 über das Moodle-System abgegeben werden.

Hausaufgabe 2.1 Daisy Chain (5 Punkte)

Das Daisy-Chain Busprotokoll wird verwendet, um mehreren Sendern den Zugriff auf ein gemeinsames Medium zu ermöglichen. Die Steuerung erfolgt einerseits durch einen Busmaster, der die Daten empfängt und über das `Grant`-Signal mitteilt, wenn er bereit für neue Daten ist. Auf der anderen Seite sind die Sender, die über ein `Request`-Signal ihren Wunsch, Daten zu versenden, mitteilen.



a) Implementieren Sie folgende Module:

- `Busmaster`: Dieses Modul empfängt die Daten der Devices und erteilt das `Grant`-Signal, wenn es bereit ist Daten zu empfangen. Es soll die folgende Schnittstelle haben:

```
module busmaster(
  input  wire        clk,
  input  wire        reset,
  input  wire        request,
  input  wire [31:0] datain,
  input  wire [3:0]  id,
  output wire        grant
)
```

- `Device`: Dieses Modul implementiert einen Sender. Die Sendedaten selbst sind in dieser Aufgabe nicht relevant. Die Schnittstelle:

```
module device #(
  parameter device_id = 0
)(
  input  wire        clk,
  input  wire        reset,
  input  wire        grant,
  output wire [31:0] dataout,
  output wire [3:0]  id,
  output wire        next_grant,
  output wire        request
)
```

- `Testbench`: Im Testmodul soll 1 Busmaster und 4 Devices instanziiert werden. Desweiteren sollen Testdaten angelegt werden, um die korrekte Implementierung zu dokumentieren.

b) Welche Probleme können bei dieser Art von Busarbitrierung auftreten? Welche Lösungen sind als Abhilfe denkbar?

Hausaufgabe 2.2 CRC-Algorithmus (5 Punkte)

Fehlerkorrektur-Algorithmen werden z.B. verwendet, um Fehler bei der Datenübertragung oder Speicherung zu erkennen (und falls möglich zu korrigieren). Ein relativ einfaches Verfahren ist das CRC-Verfahren. Hier bei wird auf der Senderseite aus den Sendedaten eine Checksumme berechnet und an die Nutzdaten angehängt. Auf der Empfängerseite wird dann überprüft, ob die Checksumme die selbe ist. Falls ja, ist (wahrscheinlich) kein Fehler bei der Übertragung passiert, falls doch wird ein Fehler gemeldet. Weitere Erläuterungen finden sich leicht durch Suche im Internet.
Für das CRC-Verfahren muss einige sowohl auf der Sende- als auch auf der Empfängerseite das verwendete Polynom gleich sein.
Wir verwenden das Polynom $x^5 + x^3 + 1$. Implementieren Sie ein Modul, das sowohl für die Generierung als auch für die Überprüfung verwendet werden kann. Das Modul soll die folgende Schnittstelle haben:

```
module crc(
  input  wire       clk,       //Takt
  input  wire       reset,     //Reset
  input  wire       datain,    //Eingang der Daten, werden seriell angelegt
  input  wire       enable,    //= 1, wenn gültige Daten am Eingang liegen
  input  wire       check,     //0 = generiere CRC, 1 = checke CRC
  output wire [4:0] dataout,   //Ausgang der Daten im Generierungsmodus
  output wire       error      //= 1, falls ein Fehler im Checkmodus festgestellt wird
  )
```

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