Einführung in Computer Microsystems

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wiederholung / Klausurvorbereitung

Klausurinfos

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hier schonmal vorab die Spielregeln bei der Klausur
- Es wird in den Teilklausuren je 60 Punkte geben
- Die Punkte werden einfach addiert (120 Punkte)
- Zum Bestehen werden 60 Punkte (50%) benötigt

1. Teilklausur

TECHNISCHE UNIVERSITÄT DARMSTADT

- Donnerstag 26.07.12, 18:00 - 19:00
- Bearbeitungszeit 60min
- Zum leichteren Verständnis/Einfinden in die Aufgabenstellung werden wir die Klausur zu Beginn vorlesen
- Während dieser Zeit sind keine Fragen erlaubt
- Während dieser Zeit ist keine Kommunikation erlaubt
- Während dieser Zeit ist kein Schreiben erlaubt

Welcher Stoff ist für die Klausur relevant

TECHNISCHE UNIVERSITÄT DARMSTADT

- Alle bisherigen Vorlesungen
- Insbesondere die Übungsblätter 4 bis 6
- ...und die Themen vom Übungsblatt 2: FSM, Busse
- Sämtliche Vorlesungen und Übungen sind nun relvant

Hilfsmittel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sie benötigen: Stift / Lineal / Getränke / Ausweis
- Ein Verilog-Syntaxblatt wird an die Klausur angehängt sein
- Ansonsten sind KEINE Hilfsmittel erlaubt

Raumeinteilung nach Nachnamen

TECHNISCHE UNIVERSITÄT DARMSTADT

- A - E: S101/A01 (70 Studenten)
- F - J: S202/C205 (63 Studenten)
- K - L: S101/A03 (45 Studenten)
- M - Z: S101/A1 (169 Studenten)

- Bitte S101/A01 und S101/A1 nicht verwechseln
- M - Z sind im großen Raum

Ergebnisse

TECHNISCHE UNIVERSITÄT DARMSTADT

- Punkte und Noten werden im Moodle eingetragen
- Bewertung wird länger dauern als 1.Teil: Korrektur, Bewertungsschema erstellen
- Anmeldung zur Klausureinsicht wird mit Notenbekanntgabe freigeschaltet
- Es wird nur einen Termin zur Einsicht geben.
- Bitte nur anmelden wenn wirklich Interesse besteht.
- Punkte feilschen hat keinen Zweck.
- Klausur ist bereits jetzt sehr fair bewertet.

HA5: Adressmap

TECHNISCHE UNIVERSITÄT DARMSTADT

In dieser Aufgabe soll ein komplettes Bussystem mit mehreren Slave-Teilnehmern aufgabaut werden. Hierzu werden zuerst einige Module erstellt und dann verbunden.
Das System soll aus folgenden Slaves bestehen:

- RAM 32KB
- ROM 4KB
- ROM 8KB
- ROM 16KB
- 8 Register zu je 32 Bit

Jede Adresse adressiert ein Byte. Alle Datenleitungen sollen 32 Bit breit sein.
Geben Sie eine gültige Adressmap an. Memory-Aliasing ist erlaubt.

Adressmap - Lösung

TECHNISCHE UNIVERSITÄT DARMSTADT

Slave	Startadresse	Endadresse
RAM 32KB	0000 0000 0000 0000	0111 1111 1111 1111
ROM 16KB	1000 0000 0000 0000	1011 1111 1111 1111
ROM 8KB	1100 0000 0000 0000	1101 1111 1111 1111
ROM 4KB	1110 0000 0000 0000	1110 1111 1111 1111
8 Register zu je 32 Bit	1111 0000 0000 0000	1111 0000 0001 1111

Adressdecoder

TECHNISCHE UNIVERSITÄT DARMSTADT

Implementieren Sie als Verilog-Modul einen Adressdecoder für einen Bus mit einer CPU als Initiator/Master. Der Decoder soll mit möglichst wenigen Gattern auskommen und für jeden der folgenden Slave-Teilnehmer ein separates `SELECT`-Signal erzeugen.

Adressdecoder - Lösung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
module decoder(
  input wire [15:0] addr,
  output wire select_ram,
  output wire select_rom1,
  output wire select_rom2,
  output wire select_rom3,
  output wire select_register);

  assign select_ram      = ~addr[15];
  assign select_rom1     =  addr[15] & ~addr[14];
  assign select_rom2     =  addr[15] &  addr[14] & ~addr[13];
  assign select_rom3     =  addr[15] &  addr[14] &  addr[13] & ~addr[12];
  assign select_register =  addr[15] &  addr[14] &  addr[13] &  addr[12];
endmodule
```

RAM

TECHNISCHE UNIVERSITÄT DARMSTADT

Implementieren Sie den RAM-Speicher als Verilog-Modul. Es soll folgende Schnittstelle besitzen:

```
module ram(
  input  wire clk,                  //Takt
  input  wire reset,                //synchroner Reset
  input  wire select,               //Select-Leitung
  input  wire we,                   //Write-Enable, =1 wenn geschrieben werden soll
  input  wire [12:0] addr,          //Adresse
  input  wire [31:0] datain,        //Schreibdaten
  output wire [31:0] dataout);      //Lesedaten
endmodule
```

RAM - Lösung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
module ram(
  input  wire clk,                  //Takt
  input  wire reset,                //synchroner Reset
  input  wire select,               //Select-Leitung
  input  wire we,                   //Write-Enable, =1 wenn geschrieben werden soll
  input  wire [12:0] addr,          //Adresse
  input  wire [31:0] datain,        //Schreibdaten
  output wire [31:0] dataout);      //Lesedaten


reg [31:0] mem [0:8191];            //4kb RAM mit 32Bit Zeilen
reg [31:0] out;
always@(posedge clk) begin
    if(we & select)                 //Schreiben
      mem[addr] <= datain;
    out <= mem[addr];
end

assign dataout = select ? out : 32'b0;
endmodule
```

Register

TECHNISCHE UNIVERSITÄT DARMSTADT

Implementieren Sie das Register-Modul in Verilog. Es soll folgende Schnittstelle besitzen:

```
module register(
  input  wire clk,                  //Takt
  input  wire reset,                //synchroner Reset
  input  wire select,               //Select-Leitung
  input  wire we,                   //Write-Enable, =1 wenn geschrieben werden soll
  input  wire [2:0] addr,           //Adresse
  input  wire [31:0] datain,        //Schreibdaten
  output wire [31:0] dataout);      //Lesedaten

endmodule
```

Register - Lösung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
module register(
  input  wire clk,                  //Takt
  input  wire reset,                //synchroner Reset
  input  wire select,               //Select-Leitung
  input  wire we,                   //Write-Enable, =1 wenn geschrieben werden soll
  input  wire [2:0] addr,           //Adresse
  input  wire [31:0] datain,        //Schreibdaten
  output wire [31:0] dataout);      //Lesedaten

integer i;

reg [31:0] mem [0:7];               //8 Register mit 32Bit
always@(posedge clk) begin
  if(reset) begin
    for (i=0;i<8;i=i+1) begin
      mem[i] <= 32'b0;
    end
  end
  else
    if(we & select)                 //Schreiben
      mem[addr] <= datain;
end

assign dataout = select ? mem[addr] : 32'b0;     //Lesen

endmodule
```

Systematischer Entwurf

TECHNISCHE UNIVERSITÄT DARMSTADT

Entwerfen Sie analog zur in der Vorlesung gezeigten Vorgehensweise zum systematischen Schaltungsentwurf eine Schaltung für ein Verfahren zur schnellen Potenzierung ganzer Zahlen mit ganzzahligen Exponenten nach folgendem Pseudo-Code:

- Eingaben Basis `base`, Exponent `exp` vorzeichenlos zu je 8 Bit
- Ausgabe Potenz `pow` ist 512 Bit breit
- Signal `start=1` startet Rechnung
- Signal `done=1` zeigt Abschluss der Rechnung an

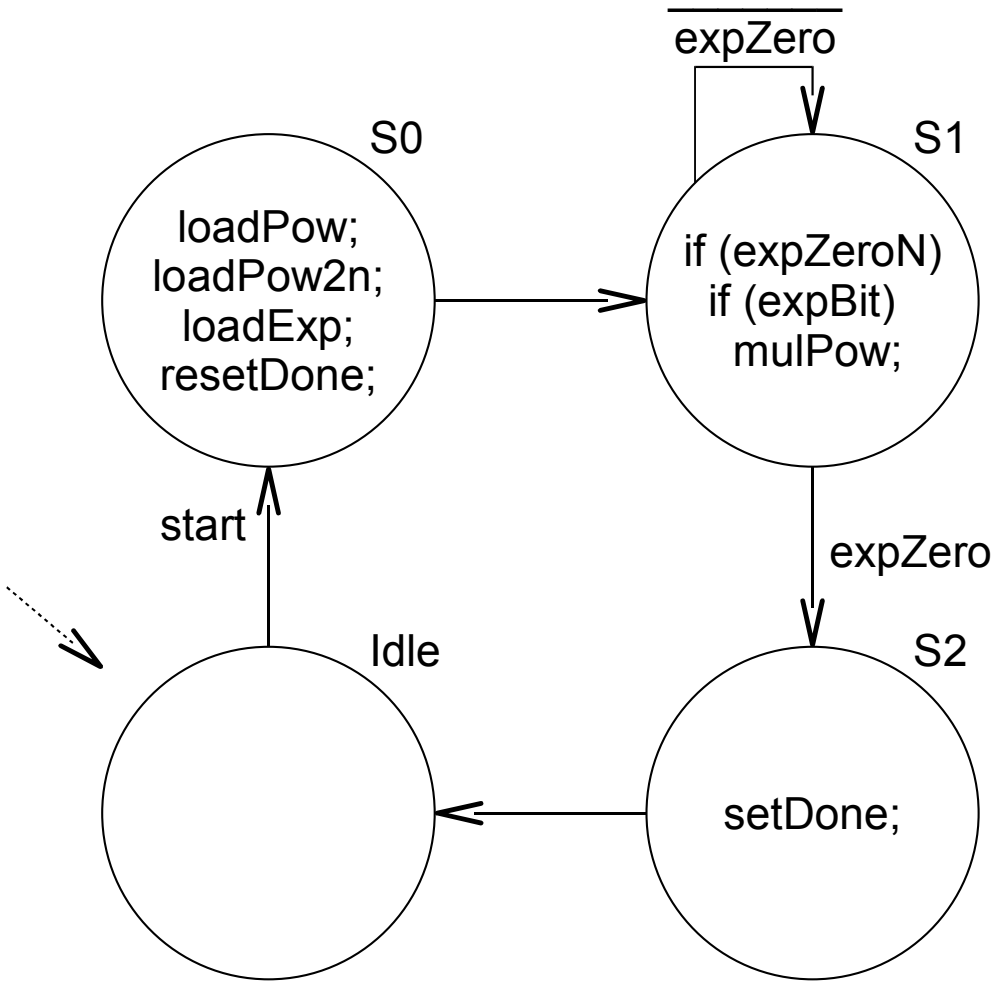
```
pow2n[511:0] := base[7:0];
pow[511:0] := 1;
done := 0;

WHILE (exp != 0) DO BEGIN
  if (exp[0] == 1)
    pow := pow * pow2n;
  pow2n := pow2n * pow2n;
  exp := exp >> 1;
END

done := 1;
```

Geben Sie den Zustandsübergangsgraph des Steuerwerks und das Diagramm des Datenpfades an.

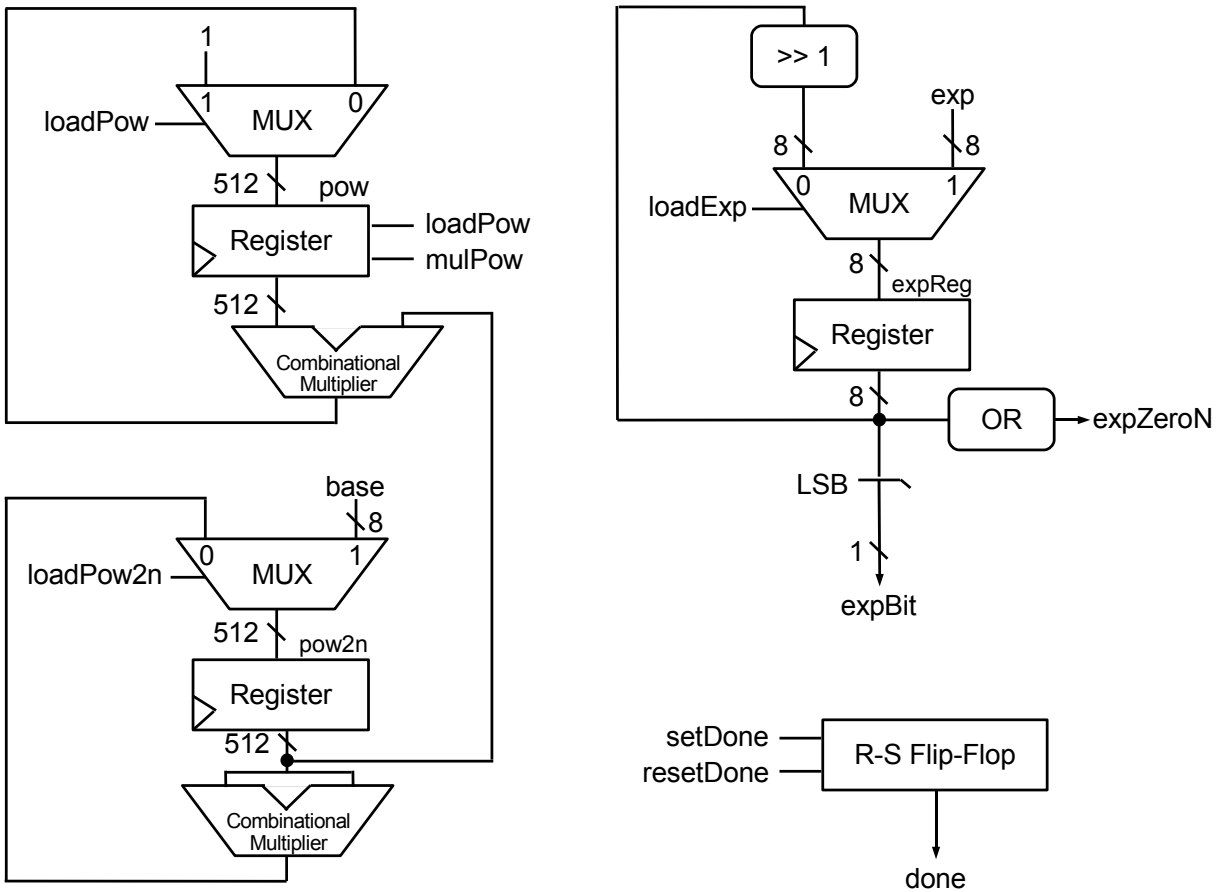
Finden Sie dazu geeignete Steuer- und Statussignale.

States - Lösung

TECHNISCHE UNIVERSITÄT DARMSTADT

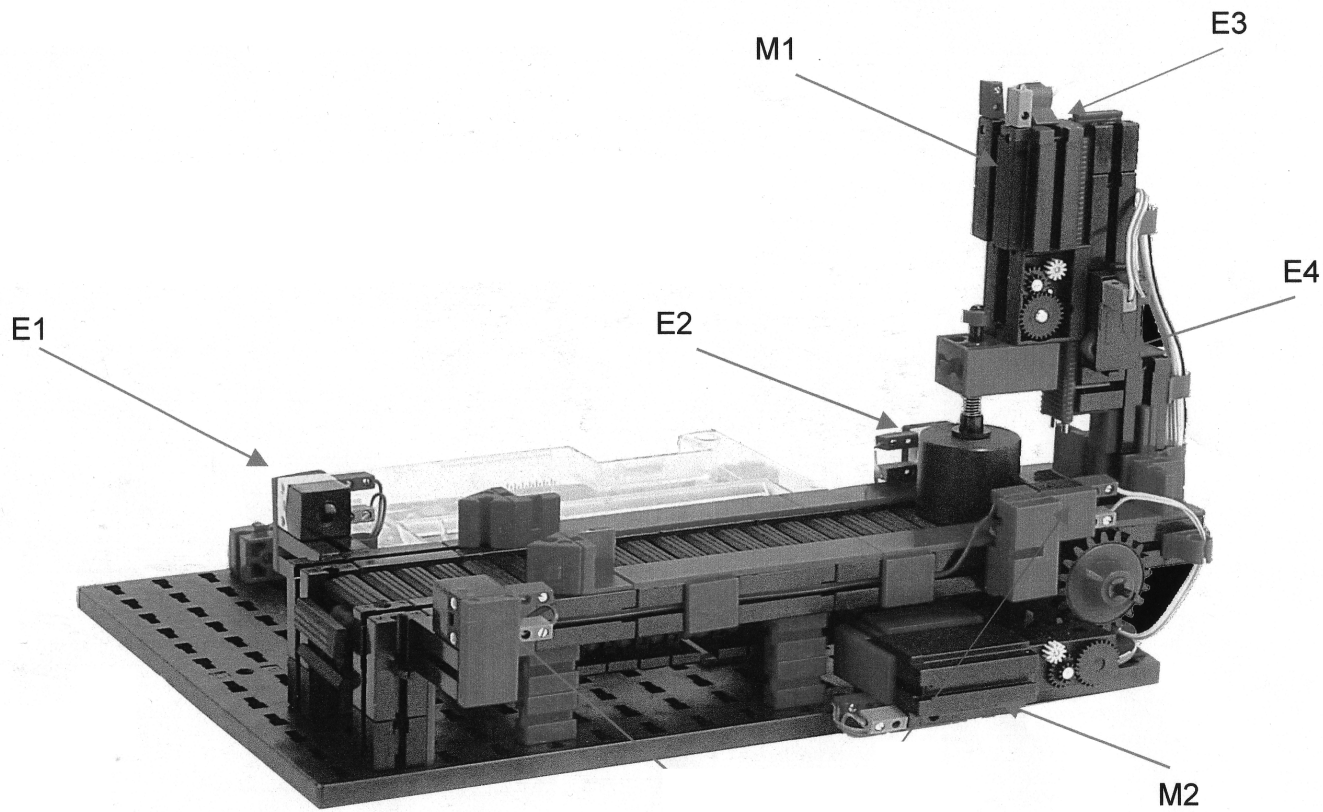


Datenpfad - Lösung



Die Stanzmaschine

TECHNISCHE UNIVERSITÄT DARMSTADT



Entwurf der Steuerung einer Stanzmaschine

TECHNISCHE UNIVERSITÄT DARMSTADT

Die zu entwerfende Steuerung soll folgende Funktion realisieren.

- Werkstück wird auf Förderband zwischen die Lichtschranke (E1) gelegt -> Das Förderband (Motor M2) wird 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.

Interface und States

TECHNISCHE UNIVERSITÄT DARMSTADT

```
module stanze(
  input wire clk,
  input wire reset,
  input wire E1, E2, E3, E4,
  output wire m1_runter,
  output wire m1_hoch,
  output wire m2_hin,
  output wire m2_zur
  );

parameter Ausgangsposition = 3'b000, Foerderband_hin     = 3'b001, Stanze_runter = 3'b010,
          Stanze_hoch      = 3'b011, Foerderband_zurueck = 3'b100;

reg [3:0] state, next_state;
```

Transitionen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
always@(*) begin
case (state)
  Ausgangsposition    : if (E1 & !E2 & !E3) next_state = Foerderband_hin;
                  else next_state = Ausgangsposition;
  Foerderband_hin     : if (E2) next_state = Stanze_runter;
                  else next_state = Foerderband_hin;
  Stanze_runter       : if (E4) next_state = Stanze_hoch;
                  else next_state = Stanze_runter;
  Stanze_hoch         : if (E3) next_state = Foerderband_zurueck;
                  else next_state = Stanze_hoch;
  Foerderband_zurueck : if (E1) next_state = Ausgangsposition;
                  else next_state = Foerderband_zurueck;
  default             : begin next_state = Ausgangsposition; end
end
```

Synchroner Block

TECHNISCHE UNIVERSITÄT DARMSTADT

```
always@(posedge clk)
begin
  if (reset == 1)
    state <= Ausgangsposition;
  else
   state <= next_state;
end

endmodule
```

Ausgaben

TECHNISCHE UNIVERSITÄT DARMSTADT

```
assign m1_runter  = (state == Stanze_runter);
assign m1_hoch    = (state == Stanze_hoch);
assign m2_hin     = (state == Foerderband_hin);
assign m2_zur     = (state == Foerderband_zurueck);
```

noch Fragen?

TECHNISCHE UNIVERSITÄT DARMSTADT

- jetzt stellen
- im Forum fragen
- Sprechstunden heute / morgen und auch Freitag
- Klausursprechstunden (nächste und übernächste Woche) werden im Moodle bekanntgegeben.

Viel Erfolg bei der Klausur!