# Einführung in Computer Microsystems
# Wiederholung / Klausurvorbereitung

TECHNISCHE
UNIVERSITÄT
DARMSTADT

# Klausurinfos

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hier schonmal vorab die Spielregeln bei der Klausur
- Es wird in beiden Teilklausuren 60 Punkte geben
- Die Punkte werden einfach addiert
- Zum Bestehen werden 60 Punkte benötigt
- Hausaufgabenbonus hilft NICHT zum Bestehen

# 2. Teilklausur

TECHNISCHE UNIVERSITÄT DARMSTADT

- 28.7.11, 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
- Relevanter Stoff: Vorlesung Kapitel 5-6, Übung 5-6
- Aber: vorheriger Stoff wird als bekannt vorausgesetzt!

# Hilfsmittel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Es wird ein Verilog-Syntaxblatt an die Klausur angehängt sein
- Ansonsten sind KEINE Hilfsmittel erlaubt

# Raumeinteilung

TECHNISCHE UNIVERSITÄT DARMSTADT

Wie erste Teilklausur: Nach Nachnamen:

- A - F: S311/08
- G - K: S101/A01
- L - Z: S101/A1

# Ergebnisse

TECHNISCHE UNIVERSITÄT DARMSTADT

- Punkte und Noten werden im Moodle eingetragen
- Bewertung wird länger dauern als 1.Teil: Korrektur, Bewertungsschema erstellen, HA-Punkte umrechnen

# 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 1100 |

# 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 [] 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 [8192:0];            //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'bz;
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


reg [31:0] mem [7:0];               //8 Register mit 32Bit
always@(posedge clk) begin
  if(reset) begin
    mem[0] <= 0;
    mem[1] <= 0;
    mem[2] <= 0;
  end
  else
    if(we & select)                 //Schreiben
      mem[addr] <= datain;
end

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

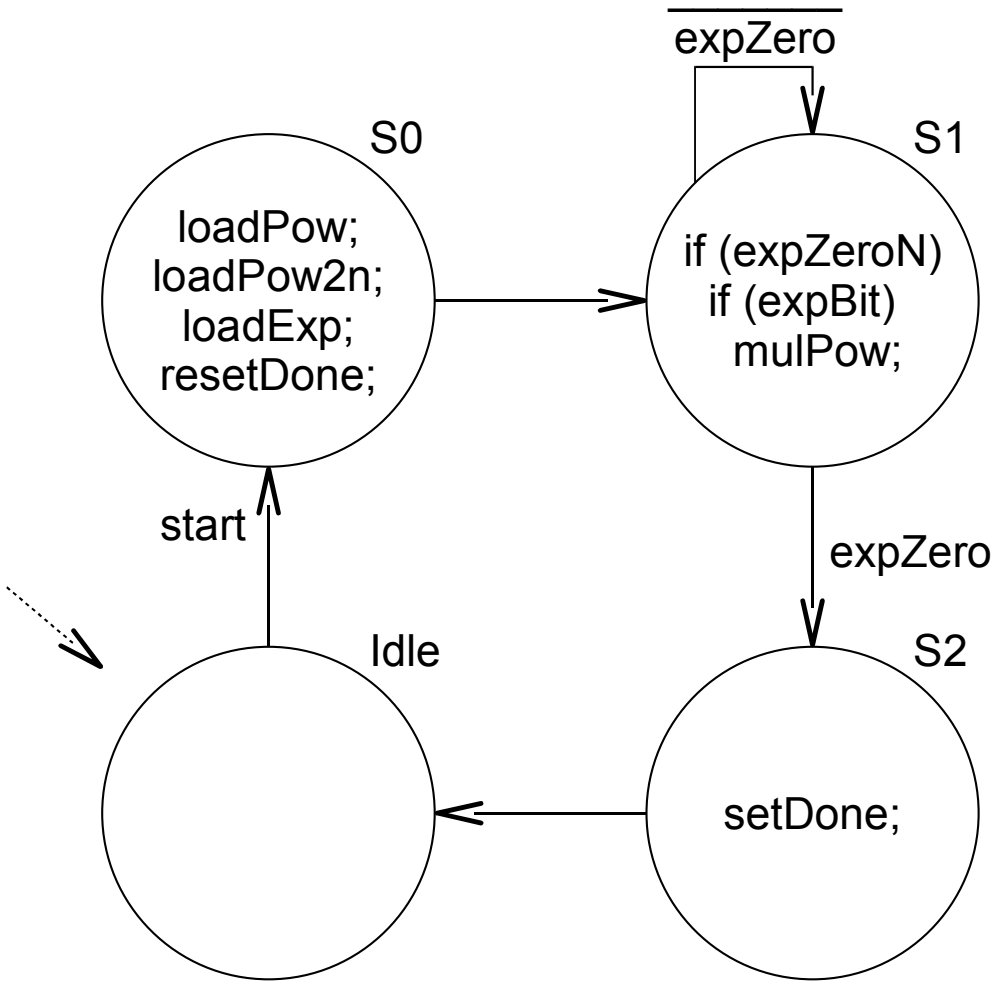
```
pow2n[511:0] := base[7:0];
pow[511:0] := 1;
done := 0;

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.

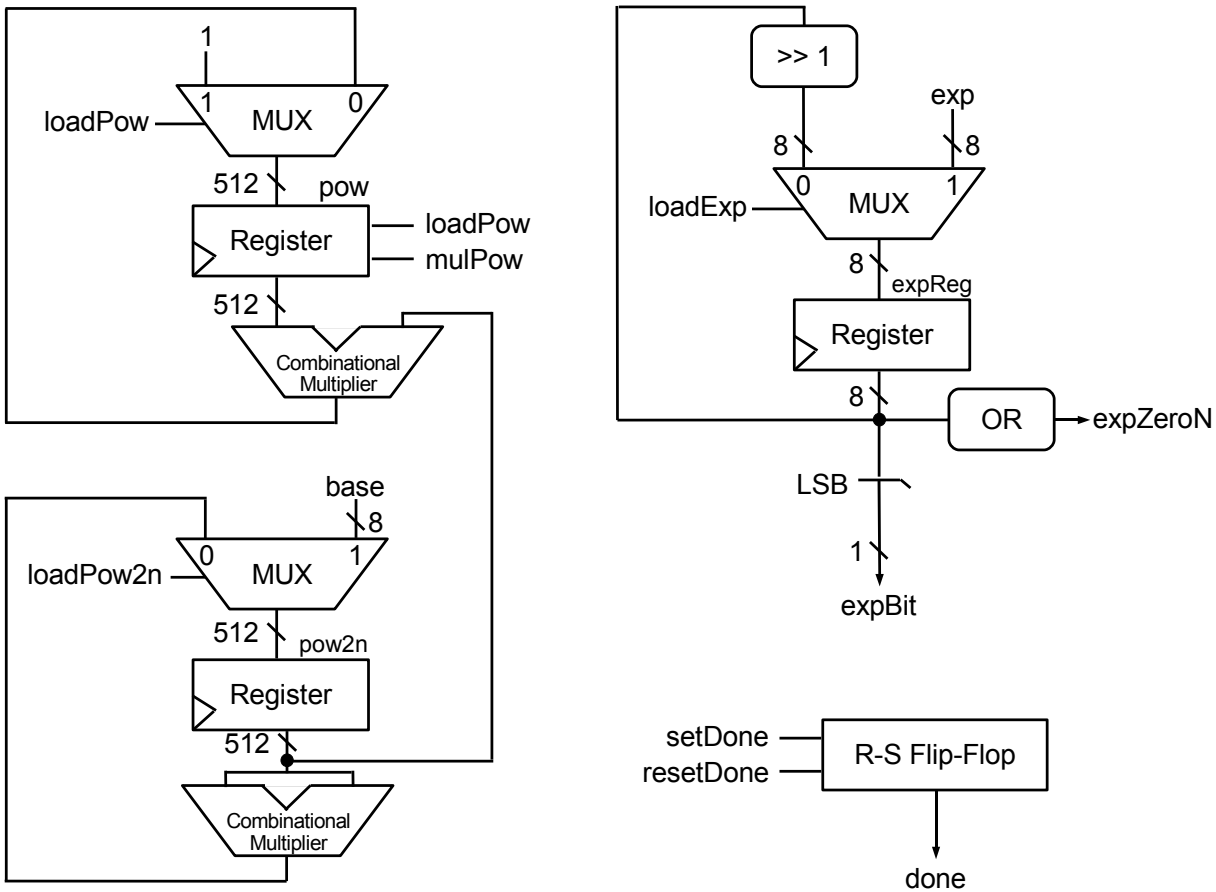
Finden Sie dazu geeignete Steuer- und Statussignale.

# States - Lösung

TECHNISCHE UNIVERSITÄT DARMSTADT



# Datenpfad - Lösung



# nach Fragen?

TECHNISCHE UNIVERSITÄT DARMSTADT

- jetzt stellen
- Sprechstunden (Termine auf Webseite/Forum)
- ins Forum stellen
- per Mail
- 
- Viel Erfolg bei der Klausur!