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

1. Teilklausur

TECHNISCHE UNIVERSITÄT DARMSTADT

- 3.6.11, 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 1-4, Übung 1-3

Hilfsmittel

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Raumeinteilung

TECHNISCHE UNIVERSITÄT DARMSTADT

Nach Nachnamen:

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

Ergebnisse

TECHNISCHE UNIVERSITÄT DARMSTADT

- Punkte werden im Moodle eingetragen
- Noten erst nach der zweiten Teilklausur

Hausaufgaben - Mittelwert

TECHNISCHE UNIVERSITÄT DARMSTADT

```
module mittelwert(
  input  wire [15:0] in1,
  input  wire [15:0] in2,
  input  wire [15:0] in3,
  input  wire [15:0] in4,
  output wire [15:0] out
  );

  //18 Bit wegen möglichem Überlauf
  wire [17:0] temp;

  // Alle vier Werte zusammen addieren.
  assign temp = in1 + in2 + in3 + in4;

  // /4 durch shiften oder direkte Auswahl der Bits
  assign res = temp[17:2];

endmodule
```

Hausaufgaben - Mittelwert

TECHNISCHE UNIVERSITÄT DARMSTADT

```
module tb()
  reg  [15:0] in1,in2,in3,in4;   //Eingänge als reg
  wire [15:0] out;               //Ausgänge als wire

  //Instanz
  mittelwert  uut (
    .in1(in1),
    .in2(in2),
    .in3(in3),
    .in4(in4),
    .out(out)
  );

  //Stimulus
  initial begin
    // alle Werte 0 --> erwartet 0
    in1 = 0;  in2 = 0;  in3 = 0;  in4 = 0; #10
    // normale Eingaben --> erwartet 10
    in1 = 10;  in2 = 5;  in3 = 5;  in4 = 20;  #10
    ..............
  end
endmodule
```

Hausaufgaben - CRC

TECHNISCHE UNIVERSITÄT DARMSTADT

```
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
  )
```

Implementierung mittels Shift-Register

Hausaufgaben - CRC

TECHNISCHE UNIVERSITÄT DARMSTADT

```
    //Shiftregister
    reg [4:0] shiftreg;

    //CRC-Polynom, kann auch unten direkt drinnen stehen
    wire [5:0] crc_poly = 6'b101001;

    always @(posedge clk) begin
      //Reset --> Shiftregister zuruecksetzen
      if(reset) begin
        shiftreg <= 6'b000000;
      end
      else begin
        //Wenn enable --> CRC berechnen
        if(enable) begin
          //wenn shiftreg[4] != dem Eingang, dann shiften und XOR
          if(shiftreg[4] != datain)
            shiftreg[4:0] <= {shiftreg[3:0], 1'b0} ^ crc_poly[4:0];
          //ansonsten nur shiften
          else
            shiftreg[4:0] <= {shiftreg[3:0], 1'b0};
        end
      end
    end

    //Ausgang zuweisen
    assign dataout = shiftreg;

    //Fehler, falls Check-Modus und Shiftregister ungleich 0
    assign error = (check & (|shiftreg));
endmodule
```

Aufgaben aus alter Klausur

TECHNISCHE UNIVERSITÄT DARMSTADT

- Folgend: Aufgaben aus alten Klausuren
- Achtung: decken nicht den gesamten Stoff ab!
- Selbst versuchen zu lösen, 10min Zeit
- Dann Diskussion der Lösung

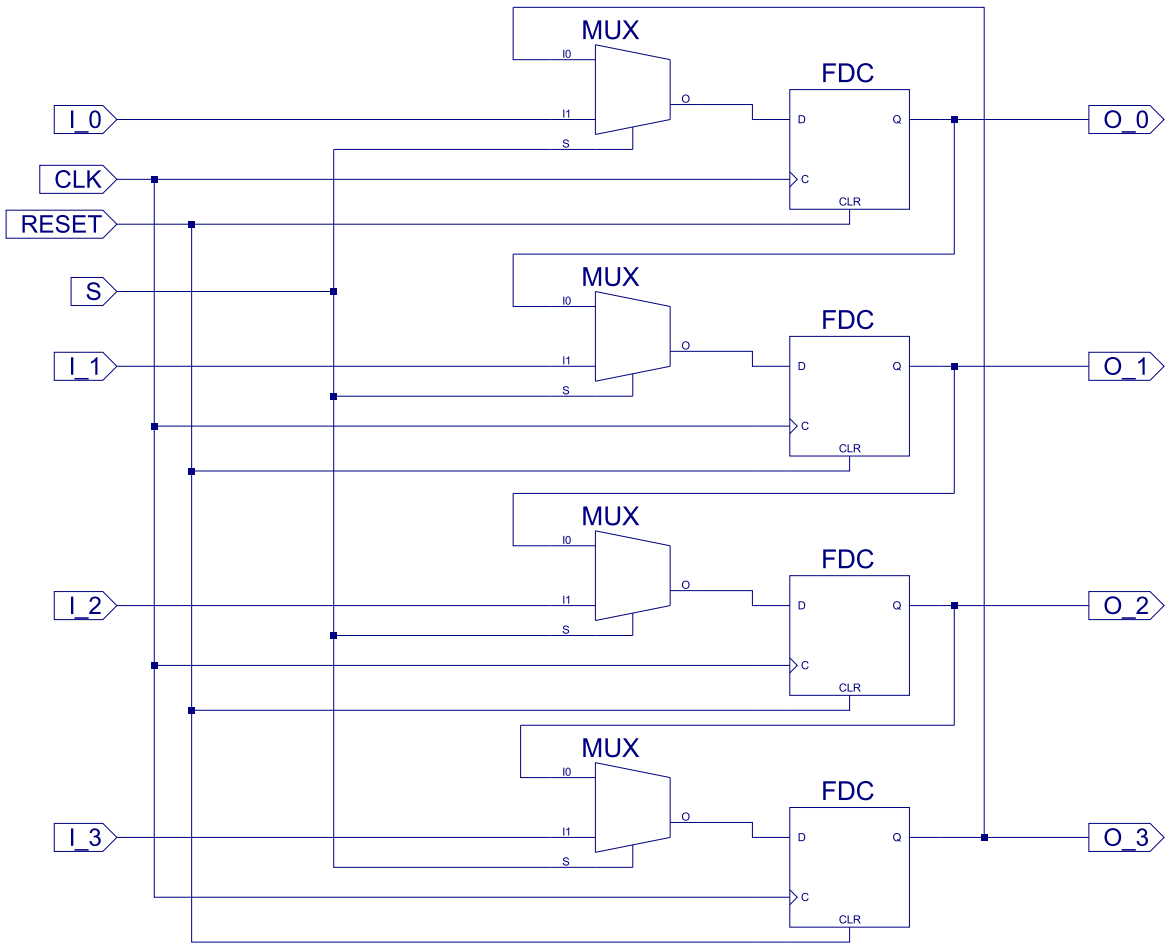
A1 - Verilog aus Schaltplan

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Konstruieren Sie ein Verilog-Modul, welches dieselbe Schaltung einschließlich aller Ein- und Ausgänge mit identischer Funktionalität implementiert. Die mit `FDC` bezeichneten Komponenten sind vorderflankengesteuerte D-Flip-Flops mit asynchronem positivem Reset.
2. Welche logische Funktion hat die Schaltung?

A1 - Schaltplan

TECHNISCHE UNIVERSITÄT DARMSTADT



A1 - Lösung

TECHNISCHE UNIVERSITÄT DARMSTADT

1.

```
1  module shiftreg(
2    input wire CLK, RESET, I_0, I_1, I_2, I_3, S,
3    output wire O_0, O_1, O_2, O_3);
4
5    //FlipFlops
6    reg [3:0] SHIFT;
7
8    //Ausgang zuweisen
9    assign {O_3, O_2, O_1, O_0} = SHIFT;
10
11   always @(posedge CLK or posedge RESET) begin
12     if (RESET)
13       SHIFT <= 0;                          //Reset
14     else if (S)
15       SHIFT <= {I_3, I_2, I_1, I_0};     // Paralleles Laden
16     else
17       SHIFT <= {SHIFT[2:0], SHIFT[3]}; // Shift
18   end
19
20 endmodule
```

2. Parallel ladbares Schieberegister/Rotationsregister

A2 - Synthese

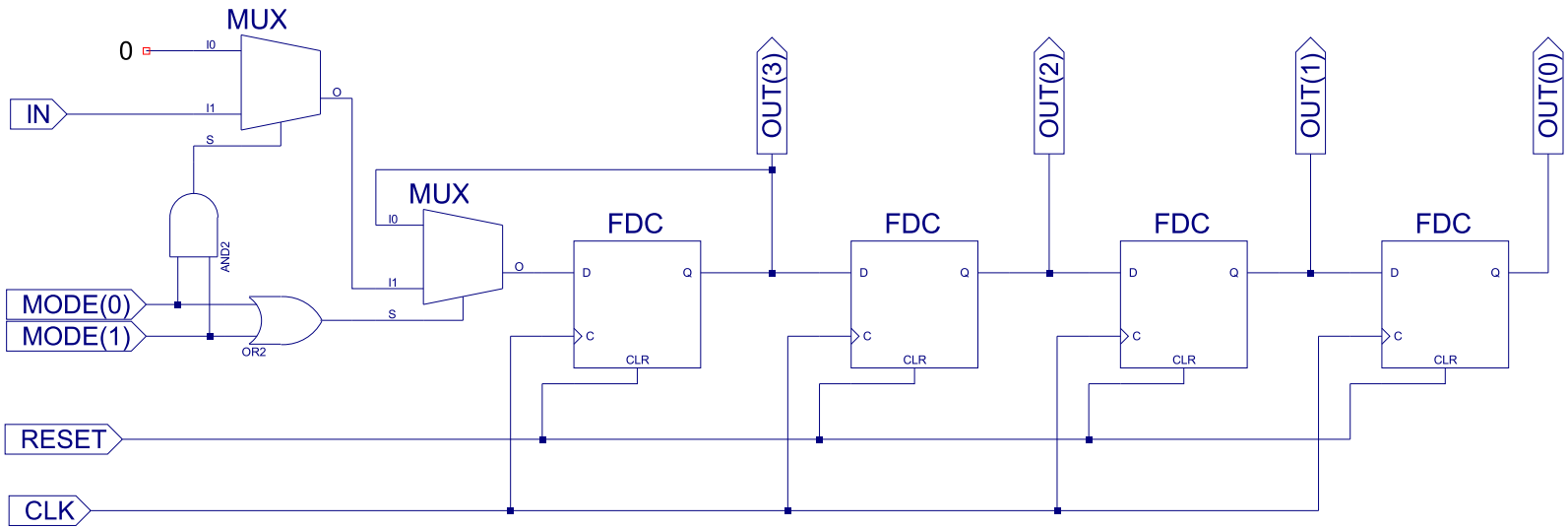
TECHNISCHE UNIVERSITÄT DARMSTADT

Geben Sie den Schaltplan auf Gatterebene für das folgende Verilog-Modul an.

```
`define ARITH 2'b00
`define LOAD 2'b11
module shift(
  input wire CLK, RESET, IN,
  input wire [1:0] MODE,
  output reg signed [3:0] OUT
);
  always @(posedge CLK or posedge RESET)
    if (RESET) OUT <= 0;
    else case (MODE)
      `LOAD:   OUT <= {IN, OUT[3:1]};
      `ARITH:  OUT <= OUT >>> 1;
      default: OUT <= OUT >> 1;
    endcase
endmodule
```

A2 - Lösung

TECHNISCHE UNIVERSITÄT DARMSTADT



A3 - Verilog

TECHNISCHE UNIVERSITÄT DARMSTADT

Gegeben ist folgender Aufwärtszähler:

```
module UpCounter(
  input wire CLK, RESET,
  output reg [3:0] COUNT
);
  always @(posedge CLK or posedge RESET)
    if (RESET) COUNT <= 0;
    else COUNT <= COUNT + 1;
endmodule
```

1. Erweitern Sie den gegebenen Zähler zu einem ladbaren Auf-/Abwärtszähler mit programmierbarer Schrittweite. Eine „1" auf dem zusätzlichen Eingang `STEP` zeigt an, dass mit der nächsten positiven Taktflanke ein Wert vom zusätzlichen Eingang `VALUE` als neue Schrittweite des Zählers geladen werden soll. Eine „1" auf dem zusätzlichen Eingang `LOAD` zeigt an, dass mit der nächsten positiven Taktflanke ein beliebiger Wert ebenso vom Eingang `VALUE` als neuer Zählerstand geladen werden soll. Bei einer „0" auf `LOAD` und `STEP` soll der Zähler bei jeder positiven Taktflanke um die programmierte Schrittweite erhöht (zusätzlicher Eingang `DOWN` := „0") bzw. erniedrigt (`DOWN` := „1") werden. Während eine neue Schrittweite geladen wird, soll der Zähler nicht zählen.
2. Der Chip aus a) soll in der Fertigung billiger werden. Welcher Pin ist redundant und kann eingespart werden? Begründen Sie ihre Entscheidung.

A3 - Lösung

TECHNISCHE UNIVERSITÄT DARMSTADT

1.

```
1  module UpDownCounter(
2    input wire CLK, RESET, STEP, LOAD, DOWN,
3    input wire [3:0] VALUE,
4    output reg [3:0] COUNT
5  );
6    reg [3:0] STEPSIZE;
7
8    always @(posedge CLK or posedge RESET)
9      if (RESET) COUNT <= 0;
10     else if (LOAD)        //neuer Zählerstand laden
11       COUNT <= VALUE;
12     else if (STEP)        //Schrittweite laden
13       STEPSIZE <= VALUE;
14     else
15       COUNT <= DOWN ? COUNT - STEPSIZE : COUNT + STEPSIZE;    //zählen
16 endmodule
```

2. Es kann nur `DOWN` eingespart werden, ohne die Funktion des Chips einzuschränken. Negative Schrittweite ist dann über Zweierkomplement der Schrittweite möglich. `RESET` ist von der reinen Logikfunktion auch denkbar, darf aber bei synchroner Logik niemals fehlen.

Themenübersicht

TECHNISCHE UNIVERSITÄT DARMSTADT

Wichtig: der Vorlesung, nicht der Klausur ;)

- Einleitung: sollte man sich nochmal durchlesen, auch Theoriefragen möglich
- Verilog Grundlagen
- Testbenches
- RTL-Pipelines
- Busse
- Automaten in Verilog
- potenzielle Register
- Synthese

Fragen

TECHNISCHE UNIVERSITÄT DARMSTADT

- for vs. generate
- Synthetisierbarkeit von /
- Werte in mehreren always-Blöcken zuweisen

nach Fragen?

TECHNISCHE UNIVERSITÄT DARMSTADT

- jetzt stellen
- Sprechstunde heute 14:00 S202/E102
- ins Forum stellen
- per Mail
-
- Viel Erfolg bei der Klausur!