Technische Universität Darmstadt
FG Eingebettete Systeme und
ihre Anwendungen (ESA)

Prof. Dr. Andreas Koch
Holger Lange

28.04.2007

Einführung in Computer Microsystems
1. Aufgabenblatt
Sommersemester 2007

Aufgabe 1: MaxMin

Gegeben ist folgende aus der Vorlesung bekannte Schaltung:

```
`timescale 1ns / 1ps
module maximum (
  input wire [7:0] A, B,
  output reg [7:0] RESULT
);

always @(A, B)
  if (A > B)
    RESULT = A;
  else
    RESULT = B;

endmodule
```

a) Legen Sie mit Xilinx ISE ein neues Projekt an, welches obiges Verilog-Modul enthält. Nutzen Sie dazu den New Project Wizard (**`File` → `New Project...`**). Geben Sie als Family/Device Virtex2P/XC2VP30 an (Package FF896, Speed -5). Erzeugen Sie eine Testbench und ergänzen sie diese, so dass alle relevanten Schaltungsfunktionen getestet werden. Verwenden Sie Verzögerungen von 1 ns (**`#1`**) zwischen den Stimuli. Führen Sie eine Verhaltenssimulation (Behavioral Simulation) durch und überprüfen Sie anhand der Waveforms die korrekte Funktion der Schaltung.

Lösung:

Testbench:

```
`timescale 1ns / 1ps
```

```
module tb_maximum_v;

    // Inputs
    reg [7:0] A;
    reg [7:0] B;

    // Outputs
    wire [7:0] RESULT;

    // Instantiate the Unit Under Test (UUT)
    maximum uut (
        .A(A),
        .B(B),
        .RESULT(RESULT)
    );

    initial begin
        // Initialize Inputs
        A = 0;
        B = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        A = 42; B = 17; #1;
        A = 16; #1;
        B = 16; #1;
    end

endmodule
```

b) Führen Sie nun eine Simulation der fertig in Hardware implementierten Schaltung aus a) (nicht der Testbench) durch. Legen Sie dazu den Schalter **`Sources for:`** oben links auf **`Post-Route Simulation`** um, aktivieren anschließend im linken mittleren Fenster den Reiter **`Processes`** und doppelklicken auf **`Simulate Post-Place & Route Model`**. Die Verilog-Beschreibung wird automatisch synthetisiert und implementiert, anschließend startet der Simulator. Vergleichen Sie die Waveforms mit denen aus a). Was fällt Ihnen auf? Was müssen Sie ändern, damit Ihre Schaltung wieder korrekt funktioniert?

Lösung:

Die Signale werden durch reales HW-Timing verzögert. Die Stimuli ändern sich zu schnell, um die Ergebnisse noch rechtzeitig berechnen zu können. Abhilfe: Langsamere Änderungen der Stimuli, z.B. durch Verzögerung **`#20`** statt **`#1`**.

c) Erweitern Sie das **`maximum`**-Modul um die wahlweise Berechnung des Minimums der Werte A und B. Ein zusätzliches Eingangssignal **`MIN`** soll die Berechnung von Max auf Min umschalten. Erweitern Sie ebenfalls die Testbench und führen Sie erneut eine Verhaltenssimulation durch. Überprüfen Sie die korrekte Funktion anhand der Waveforms.

Lösung:

Modul MaxMin:

```
`timescale 1ns / 1ps
module maxmin(A, B, MIN, RESULT);
    input wire [7:0] A;
    input wire [7:0] B;
    input wire       MIN;
    output reg [7:0] RESULT;

always @(A, B, MIN)
  if ((A > B) ^ MIN)
    RESULT = A;
  else
    RESULT = B;

endmodule
```

Zugehörige Testbench:

```
`timescale 1ns / 1ps
module tb_maxmin_v;

   // Inputs
   reg [7:0] A;
   reg [7:0] B;
   reg       MIN;

   // Outputs
   wire [7:0] RESULT;

   // Instantiate the Unit Under Test (UUT)
   maxmin uut (
      .A(A),
      .B(B),
      .MIN(MIN),
      .RESULT(RESULT)
   );

   initial begin
      // Initialize Inputs
      A = 0;
      B = 0;
      MIN = 0;

      // Wait 100 ns for global reset to finish
      #100;

      // Add stimulus here
      A = 42; B = 17; #20;
      A = 16; #20;
      B = 16; #20;
      MIN = 1;
      A = 42; B = 17; #20;
      A = 16; #20;
      B = 16; #20;
   end

endmodule
```

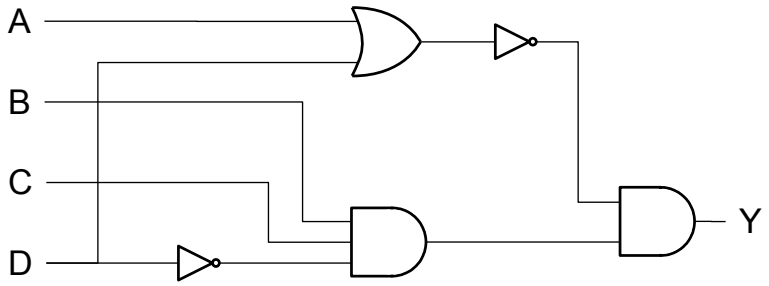
d) Führen Sie nun eine Simulation der fertig in Hardware implementierten Schaltung aus c) (nicht der Testbench) durch. Wie lange brauchen die Signale von den Eingängen **A**, **B** und **MIN** zum Ausgang **RESULT**?

Lösung:

ca. 11-14ns

Aufgabe 2: Verhaltensbeschreibung in Verilog

a) Implementieren Sie die folgende Schaltung als Verhaltensbeschreibung in Verilog. Verwenden Sie dafür einen booleschen Ausdruck.



Lösung:

```
`timescale 1ns / 1ps
module circuit(A, B, C, D, Y);
    input A;
    input B;
    input C;
    input D;
    output reg Y;

    always @(A, B, C, D) begin
      Y = ~(A | D) & (B & C & ~D);
    end

endmodule
```

b) Schreiben Sie eine Testbench für Ihr Modul aus a), die eine vollständige Wahrheitstabelle generiert. Führen Sie eine Verhaltenssimulation durch und geben Sie die Wahrheitstabelle an. Gibt es redundante Schaltungsteile? Wenn ja, welche?

Lösung:

Testbench:

```
`timescale 1ns / 1ps
module tb_circuit_v;

   // Inputs
   reg A;
   reg B;
   reg C;
```

```
        reg D;

        // Outputs
        wire Y;

        // Instantiate the Unit Under Test (UUT)
        circuit uut (
            .A(A),
            .B(B),
            .C(C),
            .D(D),
            .Y(Y)
        );

        initial begin: Stimulus
            integer i;

            // Initialize Inputs
            A = 0;
            B = 0;
            C = 0;
            D = 0;

            // Wait 100 ns for global reset to finish
            #100;

            // Add stimulus here
            for (i = 0; i < 16; i = i + 1) begin
              {D,C,B,A} = i;
              #1;
              $display("D=%b␣C=%b␣B=%b␣A=%b:␣Y=%b", D, C, B, A, Y);
            end
        end

    endmodule
```

Wertetabelle:

```
D=0 C=0 B=0 A=0: Y=0
D=0 C=0 B=0 A=1: Y=0
D=0 C=0 B=1 A=0: Y=0
D=0 C=0 B=1 A=1: Y=0
D=0 C=1 B=0 A=0: Y=0
D=0 C=1 B=0 A=1: Y=0
D=0 C=1 B=1 A=0: Y=1
D=0 C=1 B=1 A=1: Y=0
D=1 C=0 B=0 A=0: Y=0
D=1 C=0 B=0 A=1: Y=0
D=1 C=0 B=1 A=0: Y=0
D=1 C=0 B=1 A=1: Y=0
D=1 C=1 B=0 A=0: Y=0
D=1 C=1 B=0 A=1: Y=0
D=1 C=1 B=1 A=0: Y=0
D=1 C=1 B=1 A=1: Y=0
```

Redundanz:

Entweder der Inverter zwischen dem Eingang **D** und dem UND mit drei Eingängen (dann nur noch zwei Eingänge benötigt) oder das ODER sind redundant. Im letzten Fall wird der Eingang **A** direkt mit dem auf das ODER folgenden Inverter verbunden.