Digitaltechnik
Wintersemester 2017/2018
10. Übung

TECHNISCHE UNIVERSITÄT DARMSTADT

Andreas Engel, Raad Bahmani **KW02**

Die Präsenzübungen werden in Kleingruppen während der wöchentlichen Übungsstunde bearbeitet. Bei Fragen hilft Ihnen Ihr Tutor gerne weiter. Mit der angegebenen Bearbeitungszeit für die einzelnen Aufgaben können Sie Ihren Leistungsstand besser einschätzen.

Übung 10.1 Ausführungsreihenfolge [5 min]

Klammern Sie folgende Ausdrücke entsprechend der Reihenfolge der Evaluation von Teilausdrücken. Bei gleicher Präzedenz werden Operatoren von links nach rechts ausgewertet.

a) `A != B & C`

b) `A >>> B >> C`

c) `A >> B > C <<< D`

d) `A >> B + C << D`

e) `A && & B & C && D`

Übung 10.2 Verhaltens- und Strukturbeschreibung [5 min]

a) Welches der folgenden Module verwendet eine Verhaltensbeschreibung, welches eine Strukturbeschreibung?

```
module m1 (input logic  A, B, C, output logic Y);
  logic n1, n2;
  or2  i1 (A, B, n1);
  inv  i2 (.A(n1), .Y(n2));
  xor2 i3 (n2, C, Y);
endmodule
```

```
module m2 (input logic A, B, C, D, output logic Z);
  assign Z = A & ~B | ~D & C;
endmodule
```

```
module m3 (input logic A, B, C, D, output logic Y);
  logic  n1, n2, n3;
  assign n1 =  A |  B;
  assign n2 =  B &  C;
  assign n3 = n2 ~^ D;
  assign Y  = n1 ~| n3;
endmodule
```

```
module m4 (input logic A, B, C, D, output logic Y);
  nand2 i (.A(A | B), .B(((B & C) ~^ D)), .Y(Y));
endmodule
```

b) Warum dienen beide Beschreibungsarten der Abstraktion und damit der Beherrschung der Schaltungskomplexität?

Übung 10.3 Synthese [20 min]

Zeichnen Sie die von folgenden Modulen (`m1` bis `m4`) beschriebenen kombinatorischen Schaltungen.

```
module m1 (input  logic A, B,
           output logic Y);

  assign Y = A | B ? A : B;

endmodule
```

```
module m2 (input  logic [3:0] A, B,
           output logic [3:0] Y);

  assign Y = {{2{^A}},2'h3} ^ B;

endmodule
```

```
module hlp(input  logic A, B,
           output logic Y);
  assign Y = | {A,B,B,A};
endmodule

module m3 (input  logic [3:0] A,
           output logic       Y);

  logic [1:0] s;
  logic [3:0] n;
  hlp h1(n[3], n[1], s[1]);
  hlp h2(.Y(s[0]), .B(n[0]), .A(s[1]));
  assign Y = s[0] ~^ n[2];
  assign n = A << 1;

endmodule
```

```
module m4 (input  logic [1:0] A,B,C,
           output logic [1:0] Y,
           output logic       Z);

  assign Y = A &  B |  C;
  assign Z = A && B || C;

endmodule
```

Übung 10.4 Arithmetisch Logische Einheit (ALU) [20 min]

Eine ALU ist eine (kombinatorische oder sequentielle) Schaltung, welche ein Ergebnis aus mehreren Operanden berechnet. Die auszuführende Operation kann dabei über einen Selektionssignal (operation code) ausgewählt werden. Die ALU bildet damit das Herzstück der meisten Rechnerarchitekturen (siehe Vorlesung Rechnerorganisation).

Übung 10.4.1 Modul-Schnittstelle

Beschreiben Sie die Modul-Schnittstelle einer ALU mit zwei 32 bit Eingängen (`A` und `B`), einem 3 bit Selektionssignal (`OPC`), und einem 32 bit Ergebnis (`R`) mit SystemVerilog.

Übung 10.4.2 Operator-Implementierung

Die ALU soll eine Addition von `A` und `B` und einen arithmetischen Rechtsshift von `A` um `B` Stellen durchführen können. Realisieren Sie diese Operationen als SystemVerilog Module.

Übung 10.4.3 Operator-Auswahl

Implementieren Sie die ALU in SystemVerilog basierend auf den bisher beschriebenen Modulen. Für `OPC == 0` soll die Addition und für `OPC == 1` der arithmetische Rechtsshift ausgegeben werden. Für alle anderen Werte des Selektionssignals soll die ALU den Wert 0 ausgeben.

Übung 10.4.4 Operator-Erweiterung

Erweitern Sie die ALU um eine weitere Operation. Für `OPC == 2` soll `A + B + 1` berechnet werden.

Übung 10.5 Ripple-Carry Adder (RCA) [30 min]

Ein n bit RCA ist eine kombinatorische Schaltung zur Addition von zwei n bit breiten Eingängen (`A` und `B`). Das Ergebnis (`S`) der Addition ist $n+1$ bit breit. Der RCA folgt dem Prinzip der schriftlichen Addition. Beginnend bei der am wenigsten signifikanten Stelle werden die beiden Eingangsbits und das von der vorherigen Stelle übertragene carry Bit addiert. Neben der Summe für die aktuelle Stelle entsteht dabei auch der Übertrag für die nächste Stelle.
Im Folgenden soll das Zeitverhalten eines RCA mit SystemVerilog beschrieben und analysiert werden. Verwenden Sie bei allen Verhaltensbeschreibungen dafür folgende Verzögerungszeiten: $t_{\text{pd,XOR}} = 4\,\text{ns}$, $t_{\text{pd,AND}} = 3\,\text{ns}$ und $t_{\text{pd,OR}} = 2\,\text{ns}$.

Übung 10.5.1 Halbaddierer

Ein Halbaddierer ist eine kombinatorische Schaltung zur Addition von zwei 1 bit Eingängen (`A` und `B`). Das 2 bit Ergebnis wird auf die beiden Signale `S` (LSB) und `CO` (MSB) aufgeteilt. Implementieren Sie einen Halbaddierer in SystemVerilog.

Übung 10.5.2 Volladdierer

Ein Volladdierer ist eine kombinatorische Schaltung zur Addition von drei 1 bit Eingängen (`A`, `B` und `CI`). Das 2 bit Ergebnis wird auf die beiden Signale `S` (LSB) und `CO` (MSB) aufgeteilt. Implementieren Sie einen Volladdierer in SystemVerilog. Verwenden Sie dafür zwei Halbaddierer-Instanzen.

Übung 10.5.3 Carry-Chain

Implementieren Sie einen 3 bit RCA mit Hilfe von drei Volladdierern in SystemVerilog.

Übung 10.5.4 Timing-Analyse

Zeichnen Sie die kombinatorische Schaltung des 3 bit RCA. Verwenden Sie dafür möglichst wenige Basisgatter (XOR, AND und OR). Markieren Sie den kritischen Pfad der Schaltung. Mit welcher Taktrate kann die Schaltung maximal betrieben werden, wenn Operanden und Ergebnis in Registern mit folgenden Charakteristiken gespeichert werden: $t_{\text{ccq}} = 100\,\text{ps}$, $t_{\text{pcq}} = 500\,\text{ps}$, $t_{\text{setup}} = 1{,}5\,\text{ns}$, $t_{\text{hold}} = 2\,\text{ns}$? Mit welcher Frequenz könnte ein entsprechender 32 bit RCA betrieben werden?

Übung 10.5.5 Simulation

Simulieren Sie Ihren RCA mit der in Moodle bereitgestellten Testbench (`arith/rca3_tb.sv`). Warum ist die simulierte Verzögerung zwischen Ein- und Ausgang des RCA manchmal kleiner oder größer als in Übung 10.5.4 berechnet?