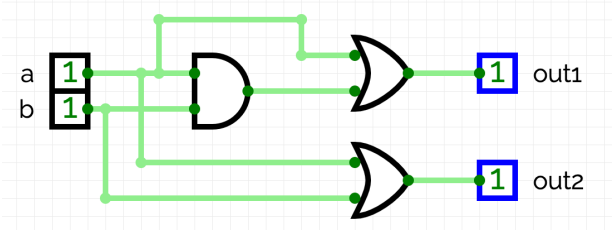
Lernziele:

- Einblick in Hardware-Beschreibungssprachen
- Formale Beschreibung von logischen Schaltungen
- Crashkurs in SystemVerilog

1. Crashkurs in SystemVerilog

Kombinatorische Logik

Wir haben gesehen, dass Schaltungen, die nur aus Logik-Gattern bestehen, durch boolesche Funktionen oder Wahrheitstabellen beschrieben werden können. Man sagt, dass sich diese Schaltungen durch "kombinatorische Logik" beschreiben lassen. Beispielsweise kann mit den Gleichungen out1 $= a \wedge b \vee a$ und out2 $= a \vee b$ folgende Schaltung beschrieben werden:



In der Praxis kommen solche mathematischen Beschreibungen in Form von HDLs (Hardware Description Languages) zur Anwendung. Zum Beispiel kann obige Schaltung in der Sprache SystemVerilog wie folgt als Modul beschrieben werden. Die logischen Gleichungen tauchen dabei im "always_comb" -Block auf:

```
module Example1 (
    input logic a,
    input logic b,
    output logic out1,
    output logic out2
);

    always_comb begin
        out1 = a & b | a;
        out2 = a | b;
    end

endmodule
```

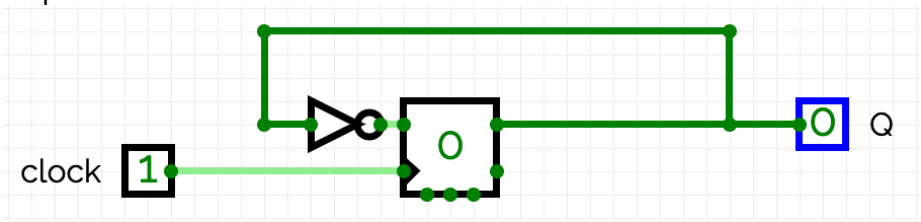
In folgendem Beispiel wurde die beschriebene Schaltung nicht geändert, sondern nur neue Bezeichnungen eingeführt:

```
module Example2 (
    input logic BTN1,
    input logic BTN2,
    output logic LED1,
    output logic LED2
);

    logic a, b;
    assign a = BTN1;
    assign b = BTN2;

    logic out1, out2;
    assign LED1 = out1;
    assign LED2 = out2;

    always_comb begin
        out1 = a & b | a;
        out2 = a | b;
    end

endmodule
```

Die Eingänge des Moduls werden jetzt "BTN1" und "BTN2" genannt, während die Ausgänge "LED1" und "LED2" genannt werden. Intern wurden diese mit assign jeweils den alternativen Bezeichnungen "a", "b", "out1" und "out2" zugewiesen. Die beschriebene kombinatorische Logik hat sich damit nicht verändert.

Sequentielle Logik

In der Vorlesung haben wir gesehen, dass bei Schaltungen mit Flip-Flops in der mathematischen Beschreibung berücksichtigt werden muss, dass diese Bauteile durch Flanken eines Clock-Signals gesteuert werden. Dies kann erreicht werden, indem man eine Folge von logischen Zuständen beschreibt. Beispielsweise kann mit den Gleichungen $Q^{(0)} = 0$ (Initialwert von Q) und $Q^{(n+1)} = \neg Q^{(n)}$(Wert von Q nach einer aufsteigenden Flanke) folgende Schaltung mit einem Flip-Flop und einem Nicht-Gatter beschrieben werden:

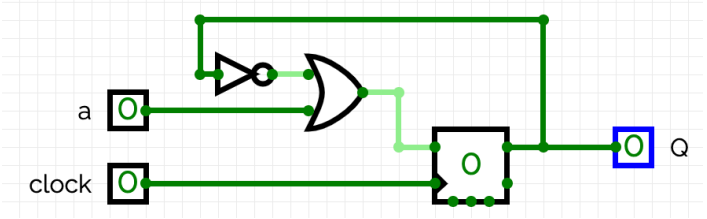


In SystemVerilog kann dies wie folgt beschrieben werden:

```
module test (
    input logic clock,
    output logic Q
);

    initial begin
        Q = 0;
    end

    always_ff @ (posedge clock) begin
        Q <= ~Q;
    end

endmodule
```

Der Initialwert wird dabei im "initial"-Block festgelegt, und wie sich der Zustand von Q nach einer aufsteigenden Flanke (positive edge) ändert, wird im "always_ff"-Block beschrieben (ff steht für Flip-Flop).

Natürlich kann in SystemVerilog ausgedrückt werden, dass eine Schaltung sowohl aus Logik-Gattern als auch Flip-Flops besteht und dementsprechend mit kombinatorischer als auch sequentieller Logik beschrieben werden kann. Beispielsweise kann



durch diesen SystemVerilog-Code beschrieben werden:

```
module test (
    input logic a,
    input logic clock,
    output logic Q
);

    initial begin
        Q = 0;
    end

    always_comb begin
        nextQ = ~Q | a;
    end

    always_ff @ (posedge clock) begin
        Q <= nextQ;
    end

endmodule
```

Aufgabe zum Knobeln:

Gegeben sei folgender SystemVerilog Code:

```
module test (
    input logic clock,
    output logic Q1,
    output logic Q2
);

    initial begin
        Q1 = 0;
        Q2 = 0;
    end

    always_comb begin
        nextQ1 = ~Q1 | Q2;
        nextQ2 = ~(Q1 & Q2);
    end

    always_ff @ (posedge clock) begin
        Q1 <= nextQ1;
        Q2 <= nextQ2;
    end

endmodule
```

Teil 1: Realisierung in CircuitVerse

Erstelle in CircuitVerse eine Schaltung, die dieser Beschreibung entspricht. Beachte, dass die Schaltung durch den SystemVerilog-Code nicht eindeutig festgelegt ist; es gibt mehr als eine mögliche Realisierung.

Teil 2: Mathematische Beschreibung

Beschreibe dieses Modul mathematisch durch Gleichungen für $Q_1^{(n)}$ und $Q_2^{(n)}$.

Teil 3: Beweis der Unmöglichkeit bestimmter Zustände

Zeige, dass unabhängig davon, wie die Initialwerte $Q_1^{(0)}$ und $Q_2^{(0)}$ gewählt werden, die Zustände $Q_1^{(n)} = 0$ und $Q_2^{(n)} = 0$ für $n > 0$ nie erreicht werden können.