SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

Systemarchitektur SS 2021

Lösungsskizze 6

Aufgabe 6.1: Speicherelemente

RS-Latch

- Ein **einzelner Puls auf /R** hat aufgrund des symmetrischen Aufbaus die entgegengesetzte Wirkung auf den RS-Latch wie ein einzelner Puls auf $/S$:

  Falls der Latch sich in dem stabilen Zustand mit $Q = 0$ befindet, verändert sich nur kurzzeitig ein Eingangssignal von G_2. Da der andere Eingang noch immer auf 0 liegt, bleibt der Ausgang von G_2 bei 1, der Latch bleibt somit im stabilen Zustand mit $Q = 0$.

  Falls der Latch wiederum in dem stabilen Zustand mit $Q = 1$ ist, verändert sich das Ausgangssignal von G_2 von 0 zu 1, damit sind beide Eingänge von G_1 auf 1. Folglich nimmt der Ausgang von G_1 0 an. Somit ist der querverbundene Eingang von G_2 ebenfalls auf 0, das Ende des Pulses auf $/R$ hat keinen Einfluss mehr auf die Schaltung. Der Latch befindet sich letztlich im stabilen Zustand mit $Q = 0$.

- Im Falle des **Puls auf /S und /R gleichzeitig** haben im ersten Schritt beide NAND-Gatter mindestens einen Eingang auf 0, damit sind beide Ausgänge 1. Durch die Querverbindungen sind die nicht gepulsten Eingänge der Gatter also auch beide auf 1.

  Bemerkenswert: In diesem Zustand gilt $Q = 1 = /Q$, also ein logischer Widerspruch zur Spezifikation; deshalb wird dies auch der "verbotene Zustand" genannt.

  Werden zu Ende des Pulses jetzt $/R$ und $/S$ beide wieder auf 1 gesetzt, nehmen die Ausgänge der NAND-Gatter jeweils 0 an. Dies trägt sich zu den querverbundenen Eingängen der Gatter weiter, woraufhin beide Ausgänge wieder 1 annehmen, wodurch sich ihre Ausgangssignale aufgrund der Querverbindungen wieder zu 0 verändern. Der Baustein würde immer weiter oszillieren, kein stabiler Zustand wird erreicht.

D-Latch

Die folgende Tabelle beschreibt die möglichen ankommenden Signale an den Eingängen $/S$ und $/R$ des im D-Latch enthaltenen RS-Latch in Abhängigkeit von der Konfiguration von den Eingängen W und D:

W	D	/S	/R
0	0	1	1
0	1	1	1
1	0	1	0
1	1	0	1

(Es gilt: $/S = \neg(W \wedge D)$ sowie $/R = \neg(W \wedge \neg D)$)

Dabei fällt auf, dass $/S$ und $/R$ in keinem Fall gleichzeitig bei 0 liegen können. Da dies jedoch für einen gleichzeitigen Puls, der die einzige Möglichkeit für das Herbeiführen eines instabilen Zustandes ist, notwendig ist, kann der RS-Latch in dem D-Latch niemals einen solchen instabilen Zustand erreichen.

Aufgabe 6.2: Taktpegel versus Taktflanken

Zusatzannahme: Das Register enthält zu Beginn den Wert 0.

1.
   - Periode t_{PQD}: An jeder Taktflanke steht der inkrementierte Wert des vorherigen Zyklus am Input D bereit. Das Register enthält also in jedem Takt den Wert des vorherigen Taktes +1. Im Zeitverlauf ergibt sich also folgende Wertefolge:
   $$0, 1, 2, 3, 4, 5, 6, 7, ...$$
   - Periode $k \cdot t_{PQD}$, $k > 1$: Da sich der Output des Registers nur an den Taktflanken ändert ist der Zustand des Schaltkreises nach dem Ablauf von t_{PQD} konstant. Ein Verlängern der Uhrperiode ändert also nichts an der Wertefolge.

2. Ersetzt man das Register durch ein Latch so bleibt das Verhalten für die Uhrperiode t_{PQD} und $2 \cdot t_{PQD}$ gleich. Erhöht man die Periode auf $k \cdot t_{PQD}$ ($k > 2$) so ändert sich der Input des Latches innerhalb der ersten Halbperiode (in denen das Uhrsignal auf 1 steht). Entsprechend ändert sich auch der Output des Latches und somit auch der Output des Inkrementers. Falls nach einem weiteren delay von t_{PQD} das Uhrsignal immer noch 1 zeigt (was ab $k = 4$ der Fall ist) so wird der neue Output des Inkrementers auch wieder in das Latch geschrieben. Es ergibt sich also als Wertefolge für $k = 4$
   $$[0, 2, 4, 6, 8, ...]$$
   oder für allgemeines $k > 1$
   $$[0, \lfloor \frac{k}{2} \rfloor, 2\lfloor \frac{k}{2} \rfloor, ...]$$

3. Falls die Propagationsverzögerung nicht konstant ist kann das Verhalten des Schaltkreises nicht mehr präzise vorhergesagt werden. Wann immer die Propagationsverzögerung Werte annehmen kann sodass das Signal innerhalb der ersten Periodenhälfte zweimal den Weg QD zurücklegt kann die Korrektheit des Schaltkreises nicht garantiert werden.

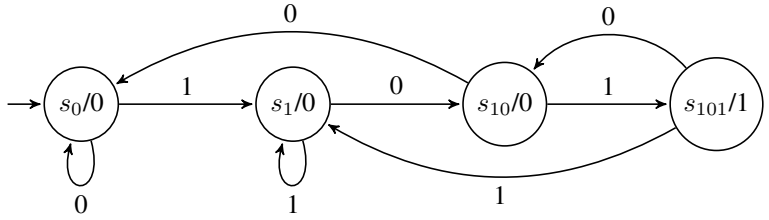
   (Anmerkung: In der Praxis hängt die Propagationsverzögerung sowohl von der Tiefe der einzelnen Schaltkreispfade als auch von äußeren Umständen wie z.B. der Temperatur ab. Die Annahme dass die Verzögerung konstant wäre ist also im Allgemeinen nicht realistisch).

Aufgabe 6.3: Schaltwerke und endliche Automaten

Beide Automaten merken sich, wie viel vom 101-Muster sie aktuell schon gelesen haben, indem sie entsprechende Zustände besuchen.

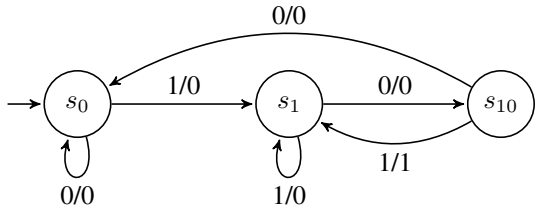
Moore-Automat

Moore mit 4 Zuständen (in den Zuständen steht die Ausgabe nach dem '/'):



Mealy-Automat

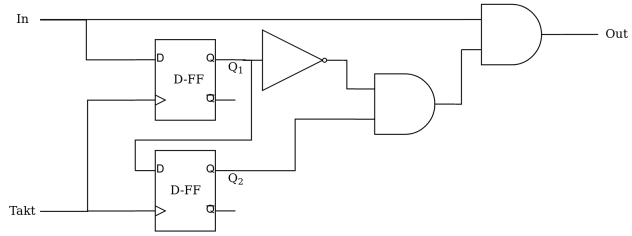
Mealy mit 3 Zuständen (Bei den Transitionen/Pfeilen steht zuerst die Eingabe, dann '/', dann die Ausgabe):



Die beiden Automaten verhalten sich unterschiedlich: Der Moore-Automat gibt eine 0 mehr am Anfang aus. Ein Moore-Automat muss erst das komplette Muster gelesen haben, um das Muster zu erkennen, da die Ausgabe nur vom Zustand abhängt. Ein Mealy-Automat kann bereits das aktuelle Eingabezeichen verwenden um das Muster zu erkennen. Daher arbeiten die beiden Automaten um einen Takt verschoben. Außer dieser Taktverschiebung um eins ist das Ein-/Ausgabeverhalten jedoch gleich.

Mealy-Schaltkreis

Wir benutzen eine etwas andere Kodierung als bei den Automaten. Mit zwei Flip-Flops speichern wir die letzten beiden Bits der Eingabe und testen, ob diese zusammen mit dem aktuellen Bit 101 ergeben:



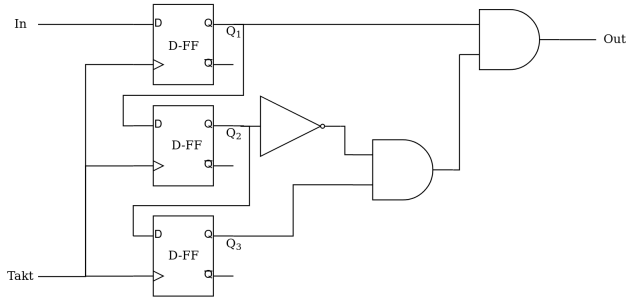
Initial sei in beiden Flip-Flops 0 gespeichert.

Verhalten:

Zyklus c	In	Q_1	Q_2	Out
0	0	0	0	0
1	0	0	0	0
2	1	0	0	0
3	0	1	0	0
4	1	0	1	1
5	0	1	0	0
6	1	0	1	1

Moore-Schaltkreis

Wir benutzen eine etwas andere Kodierung als bei den Automaten. Mit drei Flip-Flops speichern wir die letzten drei Bits der Eingabe und testen, ob diese 101 ergeben:



Initial sei in allen drei Flip-Flops 0 gespeichert.

Verhalten:

Zyklus c	In	Q_1	Q_2	Q_3	Out
0	0	0	0	0	0
1	0	0	0	0	0
2	1	0	0	0	0
3	0	1	0	0	0
4	1	0	1	0	0
5	0	1	0	1	1
6	1	0	1	0	0
6	0	1	0	1	1

Man beachte, dass es einen Schaltkreis gibt, der mit 2 Bits auskommt, die dann den $2^2 = 4$ Zuständen des Moore-Automaten entsprechen. Die Konstruktion aus der Vorlesung (mit Synthese der Übergangs-/Ausgabefunktion mittels Quine/McCluskey) wäre hier aufwändiger, würde aber ein korrektes und kompakteres Schaltwerk ergeben. Generell hängt der Aufwand dieser Konstruktion von der gewählten Kodierung der Zustände und Ein-/Ausgabe ab.

Aufgabe 6.5: Verilog: Zähler

```
module inc (
                    input [31:0] in,
                    output [31:0] out
                    );

   assign out = in + 1'b1;
endmodule
```

32-Bit incrementer

```
module counter (
    input cl,
    input clear,
    input load,
    input[31:0] X,
    output[31:0] Y
);
reg[31:0] q;
wire [31:0] incout;

initial
    begin
        q = 'b0;
    end

    inc incr(.in(q), .out(incout));

always @(posedge cl)
        begin
            if (!clear)
                q <= 'b0;
            else if (!load)
                q <= X;
            else
                q <= incout;
        end

    assign Y = q;
endmodule
```

Counter

```
module test_counter ();
    reg [31:0] in;
    reg cl, load, clear;
    wire [31:0] out;

    counter DUT (
        .cl(cl),
        .clear(clear),
        .load(load),
        .X(in),
        .Y(out)
    );

    initial
    begin
        in = 4'b1010;
        load = 1;
        clear = 1;
        cl = 0;
    end

    always
        #1
        cl = !cl;

    initial
        begin
        #50
        clear = 0;
        #2
        clear = 1;
        #8
        load = 0;
        #2
        load = 1;
    end

initial
    begin
    $dumpfile ("counter.vcd");
    $dumpvars;
    end

initial
    begin
        $display("\t\ttime ,\tcl ,\tload ,\tclear ,\tout");
        $monitor("%d ,\t%b ,\t%b ,\t%b ,\t%d",$time, cl, load, clear, out);
    end

initial
#100 $finish;

endmodule
```

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

System Architecture SS 2021

Solution Sketch 6

Problem 6.1: Memory Cells

SR Latch

- A **single pulse on /R** has the opposite effect on the SR latch as a single pulse on $/S$ due to the symmetrical structure:

  If the latch is in the stable state with $Q = 0$, only one input signal of G_2 changes temporarily. Since the other input is still at 0, the output of G_2 remains at 1, thus the latch remains in the stable state with $Q = 0$.

  If the latch is, on the other hand, in the stable state with $Q = 1$, the output of G_2 changes from 0 to 1, thus both inputs of G_1 are at 1. Consequently, the output of G_1 changes to 0. Thus, the cross-connected input of G_2 is also at 0, and the end of the pulse on $/R$ does not affect the circuit. The latch is now in the stable state with $Q = 0$.

- In the case of **pulses on /S and /R simultaneously**, in the first step both NAND gates have at least one input at 0, thus both outputs are 1. Thus, due to the cross-connections, the non-pulsed inputs of both gates are also at 1.

  Note: In this state, we have $Q = 1 = /Q$, and thus, a logical contradiction to the specification; therefore, this is also called the "forbidden state".

  Now, if $/R$ and $/S$ are both set to 1 again at the end of the pulse, the outputs of the NAND gates change to 0. This carries over to the cross-connected inputs of the gates, whereupon both outputs change to 1 again. Then, due to the cross-connections, the outputs change back to 0. The device will keep oscillating, no stable state is reached.

D Latch

The following table describes the possible signals at the inputs $/S$ and $/R$ of the SR latch (that is included in the D latch) based on the the inputs W and D:

W	D	/S	/R
0	0	1	1
0	1	1	1
1	0	1	0
1	1	0	1

(We have: $/S = \neg(W \wedge D)$ and $/R = \neg(W \wedge \neg D)$)

Note that $/S$ and $/R$ can never be at 0 at the same time. However, since this is necessary for a simultaneous pulse, which is the only way to reach an unstable state, the SR latch in the D latch can never reach such an unstable state.

Problem 6.2: Level Triggering vs. Edge Triggering

Additional assumption: The register initially contains the value 0.

1.
   - Period t_{PQD}: At each rising clock edge, the incremented value of the previous cycle is available at input D. Thus, the register contains the value of the previous cycle +1 in each clock cycle. Over time, the following sequence of values results:

     $$0, 1, 2, 3, 4, 5, 6, 7, ...$$

   - Period $k \cdot t_{PQD}$, $k > 1$: Since the output of the register changes only on rising clock edges, the state of the circuit is constant after t_{PQD} has elapsed. Thus, extending the clock period does not change the the sequence of values.

2. If the register is replaced by a latch, the behavior remains the same for the clock periods t_{PQD} and $2 \cdot t_{PQD}$. If the period is increased to $k \cdot t_{PQD}$ ($k > 2$), the input of the latch changes within the first half period (in which the clock signal is 1). Accordingly, the output of the latch also changes and so does the output of the incrementer. If after a further delay of t_{PQD} the clock signal is still at 1 (which is the case starting with $k = 4$) then the new output of the incrementer is written into the latch again. So the following sequence of values results for $k = 4$

   $$[0, 2, 4, 6, 8, ...]$$

   or for general $k > 1$

   $$[0, \lfloor\frac{k}{2}\rfloor, 2\lfloor\frac{k}{2}\rfloor, 2\lfloor]$$

3. If the propagation delay is not constant, the behavior of the circuit cannot be predicted precisely. Whenever the propagation delay can assume values such that the signal travels the path QD twice within the first half of the period, the correctness of the circuit cannot be guaranteed.

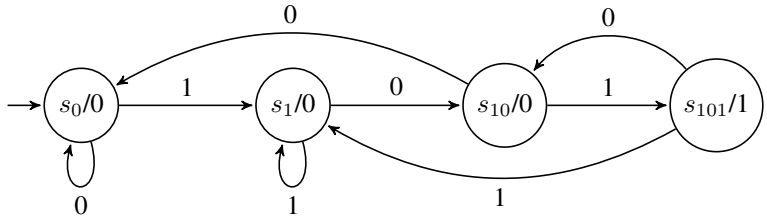
   (Note: In practice, the propagation delay depends both on the depth of the individual circuit paths and on external factors such as the temperature. Thus, the assumption that the delay is constant is, in general, not realistic).

Problem 6.3: Sequential Circuits and Finite State Machines

Both machines remember how much of the 101 pattern they have currently already read by visiting corresponding states.

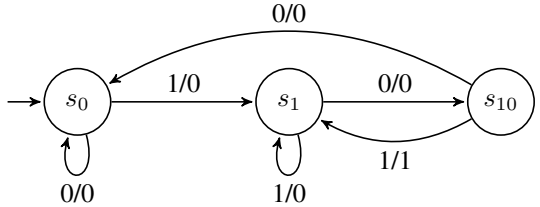
Moore Machine

Moore machine with 4 states (in the states the output is after the '/'):



Mealy Machine

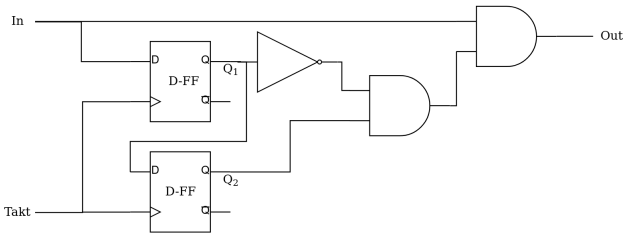
Mealy machine with 3 states (at the transitions/arrows, we first have the input, then '/', and then the output):



The two machine behave differently: The Moore machine outputs one additional 0 at the beginning. A Moore machine must first have read the complete pattern to recognize the pattern, since the output depends only on the state. A Mealy machine can already use the current input character to recognize the pattern. Therefore, the outputs of the two machines are shifted by one clock cycle. Except for this shift, however, the input/output behavior is the same.

Mealy Circuit

We use a slightly different encoding than for the machines. With two flip-flops, we store the previous two bits of the input and test whether these together with the current bit correspond to 101:



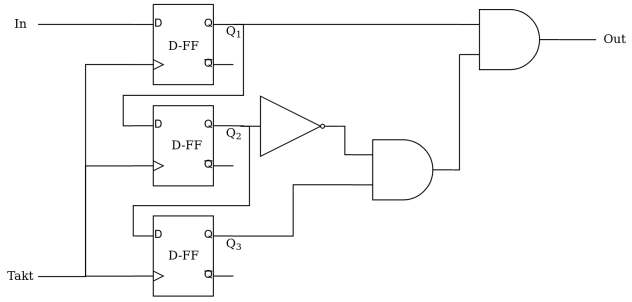
Let 0 be stored initially in both flip-flops.

Behavior:

Cycle c	In	Q_1	Q_2	Out
0	0	0	0	0
1	0	0	0	0
2	1	0	0	0
3	0	1	0	0
4	1	0	1	1
5	0	1	0	0
6	1	0	1	1

Moore Circuit

We use a slightly different encoding than for the machines. With three flip-flops, we store the previous three bits of the input and test if they correspond to 101:



Let 0 be stored initially in all three flip-flops.

Behavior:

Cycle c	In	Q_1	Q_2	Q_3	Out
0	0	0	0	0	0
1	0	0	0	0	0
2	1	0	0	0	0
3	0	1	0	0	0
4	1	0	1	0	0
5	0	1	0	1	1
6	1	0	1	0	0
6	0	1	0	1	1

Note that there is also a circuit that requires only 2 bits, which then correspond to the $2^2 = 4$ states of the Moore machine. The construction from the lecture (with synthesis of the transition/output function using Quine/Mc-

Cluskey) would require more effort, but would result in a correct and more compact sequential circuit. In general, the required effort for this construction depends on the chosen encoding of the states and the input/output.

Problem 6.5: Verilog: Counter

```
module inc (
                    input [31:0] in,
                    output [31:0] out
                    );

   assign out = in + 1'b1;
endmodule
```

32-Bit incrementer

```
module counter (
    input cl,
    input clear,
    input load,
    input[31:0] X,
    output[31:0] Y
);
reg[31:0] q;
wire [31:0] incout;

initial
    begin
        q = 'b0;
    end

    inc incr(.in(q), .out(incout));

always @(posedge cl)
        begin
            if (!clear)
                q <= 'b0;
            else if (!load)
                q <= X;
            else
                q <= incout;
        end

    assign Y = q;
endmodule
```

Counter

```
module test_counter ();
    reg [31:0] in;
    reg cl, load, clear;
    wire [31:0] out;

    counter DUT (
        .cl(cl),
        .clear(clear),
        .load(load),
        .X(in),
        .Y(out)
    );

    initial
    begin
        in = 4'b1010;
        load = 1;
        clear = 1;
        cl = 0;
    end

    always
        #1
        cl = !cl;

    initial
        begin
        #50
        clear = 0;
        #2
        clear = 1;
        #8
        load = 0;
        #2
        load = 1;
    end

initial
    begin
    $dumpfile ("counter.vcd");
    $dumpvars;
    end

initial
    begin
        $display("\t\ttime ,\tcl ,\tload ,\tclear ,\tout");
        $monitor("%d ,\t%b ,\t%b ,\t%b ,\t%d",$time, cl, load, clear, out);
    end

initial
#100 $finish;

endmodule
```