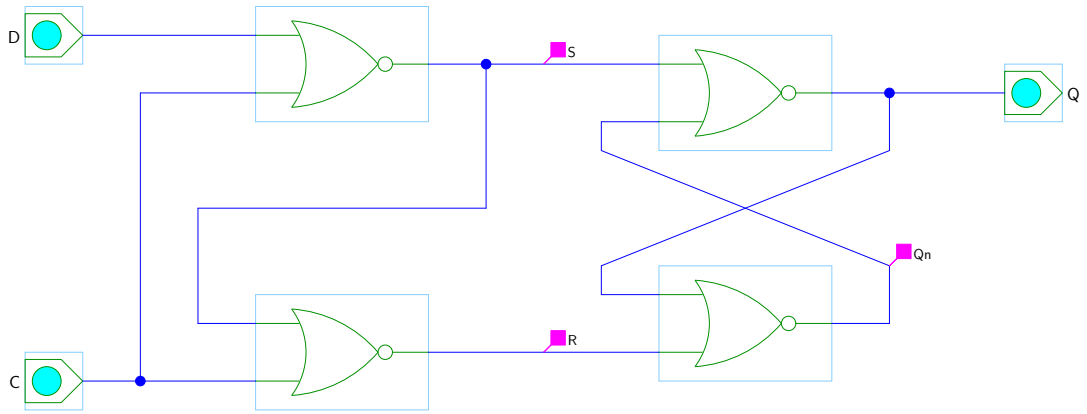
64-041 Übung Rechnerstrukturen

Aufgabenblatt 9 Ausgabe: 14.12., Abgabe: 21.12. 24:00

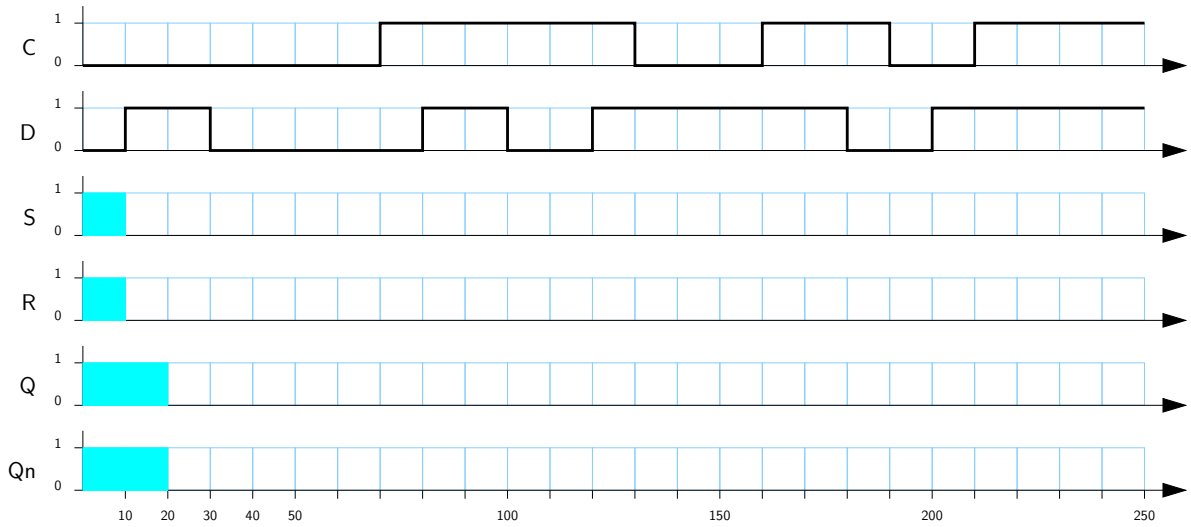
Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 9.1 (Punkte 20)

Zeitverhalten von Schaltungen: Wir untersuchen das Zeitverhalten der folgenden Schaltung mit den beiden Eingängen C und D und dem Ausgang Q.



Die Signalverzögerungszeit jedes NOR-Gatters sei genau 10 ns ist (ein Teilstrich in folgendem Diagramm). Überlegen Sie sich für die Eingaben C und D den Verlauf von S, R, Qn und Q. Beachten Sie dabei, dass wegen der Verzögerung S und R jeweils eine Zeiteinheit, Qn und Q jeweils zwei Einheiten (und möglicherweise auch noch länger) undefiniert sind. Beachten Sie dabei, dass für undefinierte Werte x gilt: $\overline{0 \vee x} = x$ und $\overline{1 \vee x} = 0$.



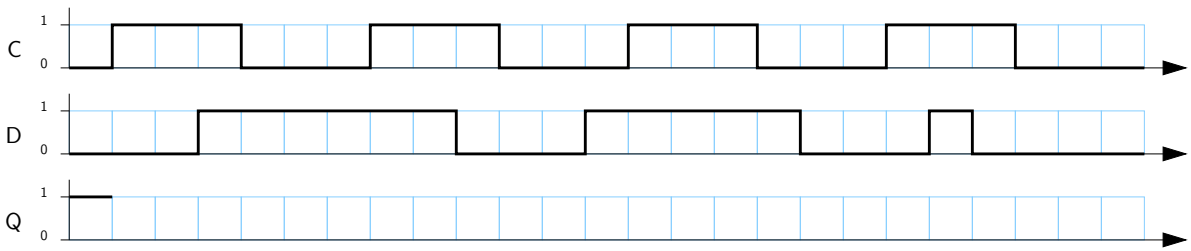
Aufgabe 9.2 (Punkte 6·5(+1))

Flip-Flop Typen: Tragen Sie zu für die folgenden Flipflops den erwarteten Signalverlauf am Ausgang Q ein. Die Flipflops sind hier in VHDL-Syntax beschrieben: `entity FF` beschreibt die Ein- und Ausgänge der Flipflops, während `architecture` ⟨*name*⟩ dann die jeweilige Implementation beschreibt. Das Verhalten jedes der Flipflops ergibt sich aus den Anweisungen im Code; `rising_edge(C)`, bzw. `falling_edge(C)` sind boole'sche Funktionen die nur wahr werden, wenn auf dem Signal `C` eine Vorder-/Rückflanke auftritt. Wenn die jeweils angegebene Bedingung für die Zuweisung (Operator <=) nicht erfüllt ist, ändert sich der Ausgang `Q` nicht.

```
library IEEE;                                -- vordefinierte Bibliothek
use IEEE.std_logic_1164.all;                 -- std_logic anstatt bit

entity FF is
port(   C       : in     std_logic;          -- Clock Eingang
        D       : in     std_logic;          -- Daten, bzw. Enable Eingang
        Q       : buffer std_logic);         -- Ausgang
end entity FF;
```

Vervollständigen Sie für jedes Flipflop das folgende Impulsdiagramm (hier *ohne Verzögerung*) und geben Sie an, um was für einen Typ es sich dabei handelt (jeweils einen Zusatzpunkt).



(a)
```
architecture ARCH92a of FF is
begin
  process(C) is
  begin
    if rising_edge(C) then
        Q <= not Q;
    end if;
  end process;
end architecture ARCH92a;
```

(b)
```
architecture ARCH92b of FF is
begin
  process(C) is
  begin
    if rising_edge(C) and (D = '1') then
        Q <= not Q;
    end if;
  end process;
end architecture ARCH92b;
```

(c)
```
architecture ARCH92c of FF is
begin
  process(C) is
  begin
    if (C = '1') then
        Q <= D;
    end if;
  end process;
end architecture ARCH92c;
```

(d)
```
architecture ARCH92d of FF is
begin
  process(C) is
  begin
    if rising_edge(C) then
        Q <= D;
    end if;
  end process;
end architecture ARCH92d;
```

(e)
```
architecture ARCH92e of FF is
begin
  process(C) is
  begin
    if falling_edge(C) then
        Q <= D;
    end if;
  end process;
end architecture ARCH92e;
```

(f)
```
architecture ARCH92f of FF is
begin
  process(C) is
    variable L : std_logic;              -- lokale Variable
  begin
    if rising_edge(C) then
        L := D;                          -- := ist Variablenzuweisung
    elsif falling_edge(C) then           -- Kurzform 'else if'
        Q <= L;
    end if;
  end process;
end architecture ARCH92f;
```

Aufgabe 9.3 (Punkte 10+10+20+10)

Entwurf eines Automaten: Zur Steuerung eines Fußgängerüberwegs soll eine Ampelschaltung entworfen werden. Beim Einschalten (Startzustand: Z_0) zeigt die Ampel für die Autofahrer grün und für die Fußgänger rot an. Durch Druck auf einen Taster ($t = 1$) wird nun eine Grünphase für den Überweg ausgelöst, ansonsten bleiben die Ampeln in Zustand Z_0 (Auto,Fußgänger = grün,rot). Wurde der Taster gedrückt, wechseln die Ampeln über Z_1 (gelb,rot) und Z_2 (rot,rot) nach Z_3 (rot, grün). Die Grünphase der Fußgänger soll 3 Takte andauern und umfasst damit 3 Zustände (Z_3 bis Z_5). Anschließend wird die Straße wieder auf Grün geschaltet und es werden die Zustände Z_6 (rot,rot), Z_7 (rot+gelb,rot) und Z_8 (grün,rot) durchlaufen. Die Grünphase der Autos dauert mindestens zwei Takte: Z_8 und dann erneut Z_0. Der Taster t ist ausschließlich im Zustand Z_0 wirksam.

(a) Zeichnen Sie das Zustandsdiagramm des Moore-Automaten.

(b) Ergänzen Sie die fehlenden Zustände und die zugehörigen Ausgangswerte. Die Tabelle enthält links den Eingangswert t und den aktuellen Zustand Z in Binärcodierung (z_3, z_2, z_1, z_0). Angegeben sind dann der Folgezustand Z^+ und die Ausgangswerte zum Ansteuern der Lampen von Autoampel (A_{rt}, A_{ge}, A_{gr}) und Fußgängerampel (F_{rt}, F_{gr}). Verwenden Sie bei Bedarf *don't-care* Werte.

t	z_3	z_2	z_1	z_0	z_3^+	z_2^+	z_1^+	z_0^+	A_{rt}	A_{ge}	A_{gr}	F_{rt}	F_{gr}
0	0	0	0	0					0	0	1	1	0
1	0	0	0	0					0	0	1	1	0
$*$	0	0	0	1	0	0	1	0					
$*$	0	0	1	0									
		• • •					• • •						

(c) Übertragen Sie die Funktionen der Zustandstabelle in KV-Diagramme und minimieren Sie die einzelnen Funktionen. Markieren Sie mögliche Schleifen und geben Sie die zugehörigen Ausdrücke für Folgezustand und Ausgangswerte in disjunktiver Form an.

(d) Erweitern Sie den Automaten so, dass sich die Ampel nach einiger Zeit ausschaltet: wenn der Taster zwei Takte (nach Erreichen von Z_0) nicht gedrückt wurde. Dazu können die „freien" Codierungen Z_9 bis Z_{15} genutzt werden. Beschreiben Sie (textuell) die Funktionsweise und zeichnen Sie das zugehörige Zustandsdiagramm.