georg.von-der-brueggen [☺] tu-dortmund.de
harun.teper [☺] tu-dortmund.de

Hardwarepraktikum
Wintersemester 21/22

Übungsblatt 5 – 15 Punkte

(Block B1 – insgesamt 70 Punkte)

Bearbeiten ab Samstag, 20. November 2021.
Abgabe bis spätestens Freitag, 26. November 2021, 23:59 Uhr.

5.1 Sequentielle Schaltungen (8 Punkte)

In der letzen Übung wurden kombinatorische Schaltungen (auch *Schaltnetze* genannt) implementiert und analysiert. Hierbei sollte aufgefallen sein, dass die Ausgabe von kombinatorischen Schaltungen nur von den aktuellen Eingabewerten abhängt und vorherige Eingaben keinen Einfluss auf die Ausgabe haben. In dieser Übung sollt ihr einige *sequentielle Schaltungen* (auch *Schaltwerke* genannt) entwerfen und implementieren. Sequentielle Schaltungen sind komplizierter als kombinatorische Schaltungen, da ihre Ausgabe nicht nur von den aktuellen Eingabewerten sondern auch von vorherigen Eingaben abhängt.

5.1.1 SR-Latch in VHDL

Eine der grundlegendsten sequentiellen Schaltungen ist das *SR-Latch*, dessen Schaltbild und Wahrheitstabelle in Abbildung 1 bzw. in Tabelle 1 dargestellt sind. Die Funktionsweise des SR-Latches kann aus dem Schaltbild und der Wahrheitstabelle abgelesen werden, insgesamt ergeben sich hierbei vier verschiedene Fälle:

- Fall I: R=1, S=0

  Wenn R den logischen Wert 1 besitzt, wird das NOR-Gatter N_1 die Ausgabe 0 am Ausgang Q erzeugen. Da sowohl Q als auch S den logischen Wert 0 haben, produziert das NOR-Gatter N_2 die Ausgabe 1 am Ausgang $\overline{Q}$.

- Fall II: R=0, S=1

  Wenn S den logischen Wert 1 besitzt, wird das NOR-Gatter N_2 die Ausgabe 0 am Ausgang $\overline{Q}$ erzeugen. Da sowohl $\overline{Q}$ als auch R den logischen Wert 0 haben, produziert das NOR-Gatter N_1 die Ausgabe 1 am Ausgang Q.

- Fall III: R=1, S=1

  Wenn R und S den logischen Wert 1 besitzen, produzieren die NOR-Gatter N_1 und N_2 die Ausgabe 0 an beiden Ausgängen Q und $\overline{Q}$.

- Fall IV: R=0, S=0

  Wenn R und S den logischen Wert 0 besitzen, sind die Ausgaben von N_1 und N_2 abhängig von den vorherigen Werten von Q und $\overline{Q}$, die als Q_{prev} und $\overline{Q}_{prev}$ angegeben sind.

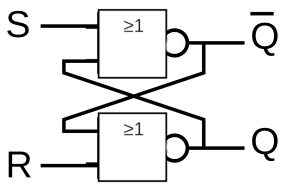

Abbildung 1: Schaltbild eines SR-Latches.

Fall	S	R	Q	$\overline{Q}$
IV	0	0	Q_{prev}	$\overline{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

Tabelle 1: Wahrheitstabelle eines SR-Latches.

Die Signale S und R des SR-Latches können also verwendet werden, um die Ausgabe Q (und dessen Negation $\overline{Q}$) zu setzen (*set*) und zu löschen (*reset*). Q wird auf den Wert 0 gesetzt wenn an R der logische Pegel 1 angelegt wird. Wenn an S der logische Pegel 1 angelegt wird, wird Q auf den Wert 1 gesetzt. Ein SR-Latch gleichzeitig setzen und löschen zu wollen ist nicht sinnvoll, in diesem Fall (wenn also an beiden Eingängen S und R der logische Pegel 1 angelegt wird) befindet sich der Wert 0 an beiden Ausgängen Q and $\overline{Q}$. Dieser Zustand wird auch *irregulärer Zustand* genannt. Wenn an beiden Eingängen S und R der logische Pegel 0 angelegt wird, speichert das SR-Latch die vorherigen Werte von Q and $\overline{Q}$, also Q_{prev} und $\overline{Q}_{prev}$. **Aufgaben:**

a. (2 Punkte) Ihr habt in Übung 2 gelernt, wie man ein Logikgatter baut (AND, OR, NOT). Entwerft ein $\overline{SR}$-Latch mit Logikgatter gemäß der Wahrheitstabelle 2. Zeichnet das Schaltbild eures entworfenen $\overline{SR}$-Latches und implementiert es in VHDL.

b. (1 Punkte) Testet eure Implementierung mit allen Kombinationen von S und R (00, 01, 10 und 11).

c. (1 Punkte) Erklärt die Nachteile beziehungsweise allgemeinen Probleme von SR-Latches.

Fall	S	R	Q	$\overline{Q}$
IV	1	1	Q_{prev}	$\overline{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	0	0	1	1

Tabelle 2: Wahrheitstabelle eines $\overline{SR}$-Latches.

5.1.2 D-Latch in VHDL

Ein weiterer häufiger Latch-Typ ist das *D-Latch*, das im Grunde genommen eine Erweiterung des SR-Latches ist. Wie schon das SR-Latch besitzt das D-Latch zwei Dateneingänge, wenn auch mit anderer Bedeutung: einen Dateneingang D und einen Takteingang CLK. Der Dateneingang D bestimmt den nächsten Ausgabewert des Latches, wohingegen der Takteingang kontrolliert, wann sich die Ausgabe ändern soll. Das Schaltbild und die Wahrheitstabelle sind in Abbildung 2 und Tabelle 3 dargestellt. Unter Zuhilfenahme dieser kann nachvollzogen werden, wie das D-Latch funktioniert:

Wenn an CLK der Logikpegel 0 anliegt, erzeugen die beiden AND-Gatter den Wert 0 an ihren Ausgängen und somit auch an S und R des verwendeten SR-Latches, was bedeutet, dass das D-Latch in diesem Fall die vorherigen Werte Q_{prev} und $\overline{Q}_{prev}$ unabhängig vom an D anliegenden Wert speichert. Wenn an CLK der Logikpegel 1 anliegt, wird der an D anliegende Wert gespeichert.

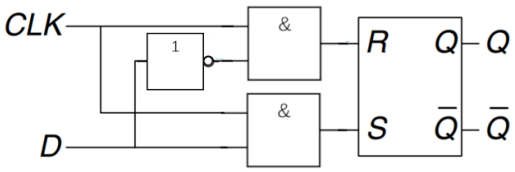

Abbildung 2: Schaltbild eines D-Latches.

CLK	D	Q	$\overline{Q}$
0	x	Q_{prev}	$\overline{Q}_{prev}$
1	0	0	1
1	1	1	0

Tabelle 3: Wahrheitstabelle eines D-Latches.

Aufgaben:

a. (2 Punkte) Entwerft ein $\overline{D}$-Latch basierend auf der Wahrheitstabelle 4. Zeichnet das Schaltbild des $\overline{D}$-Latches und implementiert es in VHDL.

b. (1 Punkte) Testet eure Implementierung mit allen Kombinationen von CLK und D (00, 01, 10 und 11).

c. (1 Punkte) Erklärt den Vorteil des D-Latches im Vergleich zum SR-Latch.

CLK	D	Q	$\overline{Q}$
0	x	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0
1	1	0	1

Tabelle 4: Wahrheitstabelle eines $\overline{D}$-Latches.

5.2 Flip-Flops in VHDL (7 Punkte)

Zusätzlich zu den vorherigen Schaltwerken ist das D-Flip-Flop eine weitere grundlegende sequentielle Schaltung, die in der Lage ist, einen Wert zu speichern. Der Unterschied zwischen dem D-Flip-Flop und dem D-Latch ist, dass der Ausgang Q des D-Latches sich zu jeder Zeit ändern kann (sofern an EN der logische Pegel 1 anliegt), wohingegen sich beim D-Flip-Flop der Wert am Ausgang Q nur zum Zeitpunkt einer aufsteigenden Taktflanke ändern kann. Zu allen anderen Zeitpunkten wird der Ausgabewert gespeichert.

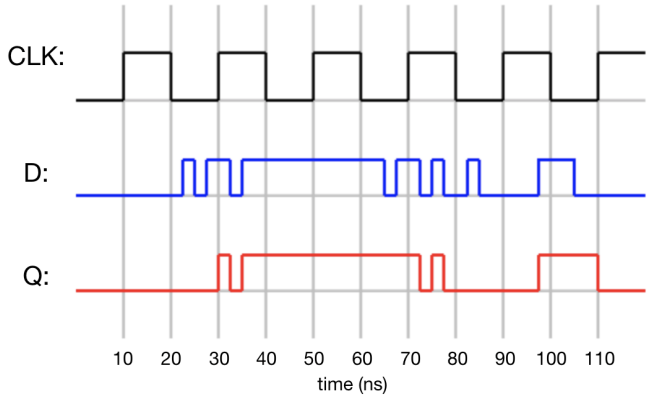

Abbildung 3: D-Latch Signalverlauf.

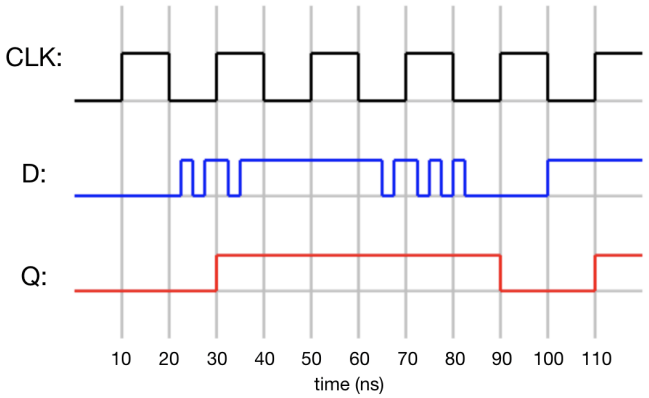

Abbildung 4: D-Flip-Flop Signalverlauf.

Um diesen Unterschied besser zu verstehen, ist ein Beispielszenario in den Abbildungen 3 und 4 dargestellt, bei dem der Signalverlauf für ein D-Latch, respektive D-Flip-Flop gezeigt wird. Wie in Abbildung 3 zu sehen ist, ändert sich die Ausgabe des D-Latches nach 32 ns, weil der Eingabewert von D sich zu diesem Zeitpunkt ändert. Das D-Flip-Flop ändert seine Ausgabe zu diesem Zeitpunkt jedoch nicht, da keine aufsteigende Taktflanke vorliegt. Das D-Flip-Flop kann wie in Abbildung 5 dargestellt aus zwei hintereinandergeschalteten D-Latches mit komplementärem Taktsignal gebaut werden.

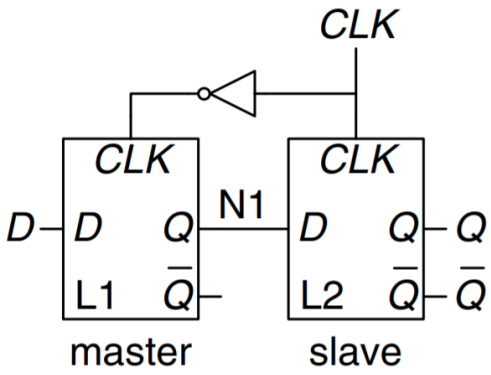

Abbildung 5: Schaltbild eines Master-Slave-D-Flip-Flops

Aufgaben:

a. (1 Punkte) Erklärt ausführlich, wie das Master-Slave-D-Flip-Flop in Abbildung 5 funktioniert.

b. (2 Punkte) Implementiert ein D-Flip-Flop basierend auf der Wahrheitstabelle 5 und testet eure Implementierung mit unterschiedlichen Kombinationen von CLK und D.

(Hinweis: Ihr könnt "***wait on** signal **until** condition*" verwenden, um das Verhalten von Master-FF und Slave-FF zu simulieren. Für das Master FF ist der Ausgang Q_m gleich D_m sobald "***wait on** CLK **until** CLK=1*" erfüllt ist. Für das Slave FF ist der Ausgang Q_s gleich D_s sobald "***wait on** CLK **until** CLK=0*" erfüllt ist.)

CLK	D	Q	$\overline{Q}$
0	x	Q_{prev}	$\overline{Q}_{prev}$
1	0	0	1
1	1	1	0

Tabelle 5: Wahrheitstabelle eines $\overline{D}$-Flip-Flops.

c. (1 Punkte) Damit ein Flip-Flop in einen definierten Startzustand gebracht werden kann, muss es ermöglicht werden, dieses zurücksetzen zu können (*Reset*). Erklärt den Unterschied zwischen Flip-Flops mit synchronem und asynchronem Reset.

d. (2 Punkte) Erweitert das Flip-Flop, das ihr in Aufgabenteil b entworfen habt, um die Funktion eines synchronen Resets. Zeichnet das Schaltbild dieses synchronen und resetbaren Flip-Flops und implementiert es in VHDL.

e. (1 Punkte) Testet eure Implementierung mit allen Kombinationen von CLK und D (00, 01, 10, 11), danach setzt eures Flip-Flop auf den Startzustand (00) zurück.