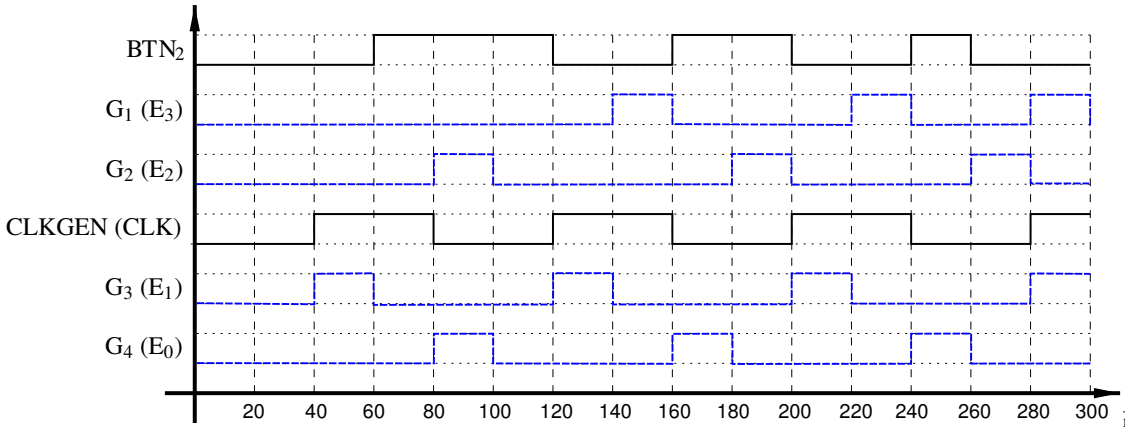
1 Messungen von Verzögerungs- und Anstiegszeiten

1.1 Bedeutung der Gatter $G_1 - G_4$

G_1 erkennt eine fallende Flanke von BTN_2 (also ein losgelassener Schalter).
G_2 erkennt eine steigende Flanke von BTN_2 (also ein gedrückter Schalter).
G_3 erkennt eine steigende Flanke von CLKGEN.
G_4 erkennt eine fallende Flanke von CLKGEN.

1.2 Signalverlauf für die Gatter $G_1 - G_4$



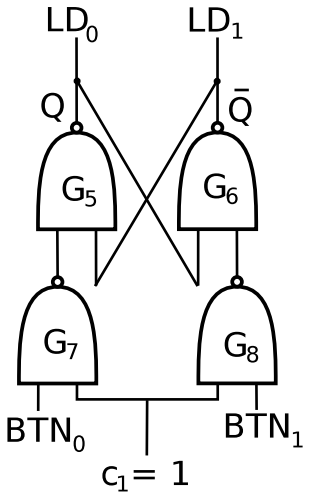
1.3 interner Aufbau der 2 RS-Latches

Während bei einem Active-High-NOR-RS-Latch beide Ausgänge auf low schalten, wenn beide Eingänge auf high gesetzt sind, schalten in diesem Fall beide Ausgänge beim Active-High-NAND-RS-Latch auf high.
Eine alternative Idee wäre folgende:
Um aus einem Active Low NAND-RS-Latch einen Active High NAND-RS-Latch zu machen, muss man vor die beiden Eingänge $\overline{S}$ und $\overline{R}$ einen Inverter setzen. Damit dauert es bei dem Active High NAND-RS-Latch länger als beim NOR-RS-Latch bis die Ausgänge stabil sind, wenn man davon ausgeht dass die Gatterlaufzeiten von NOR und NAND gleich sind. Diese Idee basiert allerdings auf der Vermutung, dass NAND und NOR die gleiche Verzögerung mit sich bringen.

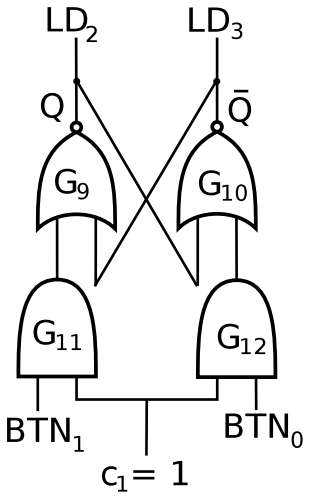
1.4 Erzeugung zweier RS-Latches aus $G_5 - G_{12}$

1.4.1 Active-High NAND-RS-Latch

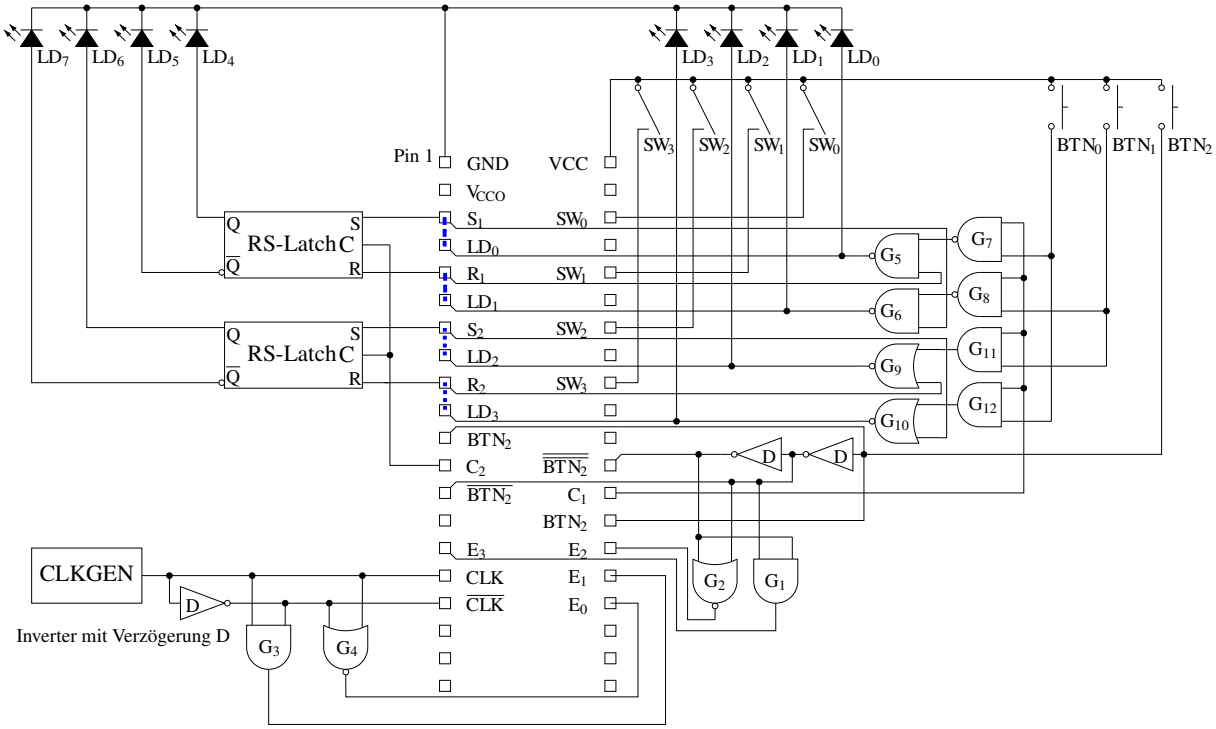
Aus G_5 und G_6 kann durch Verbinden von S_1 mit LD_0 und R_1 mit LD_1 ein Active-Low NAND-RS-Latch erzeugt werden.
Durch Vorschalten von G_8 und G_7 kann mit BTN_0 ein Set und mit BTN_1 ein Reset durchgeführt werden.
Dafür muss C_1 high sein.
LD_0 entspricht hier unsrem Q, LD_1 unsrem $\overline{Q}$.



1.4.2 (Active-High) NOR-RS-Latch

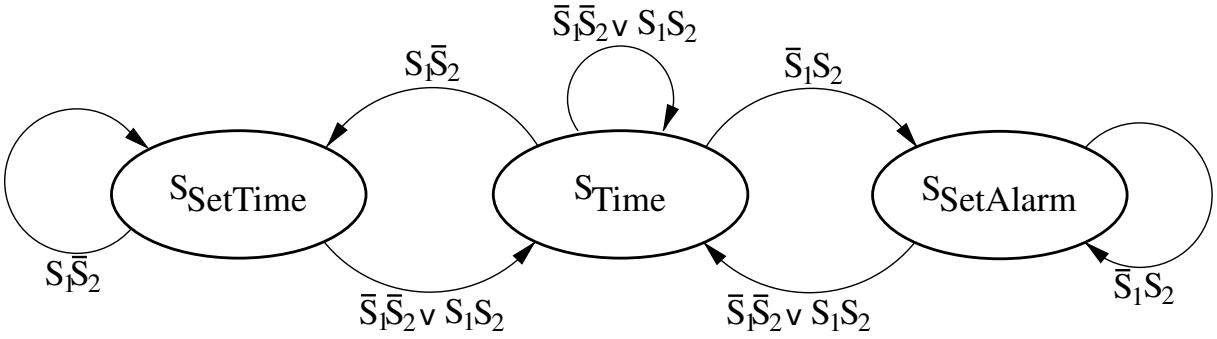


Aus G_9 und G_{10} kann durch Verbinden von S_2 mit LD_2 und R_2 mit LD_3 ein NOR-RS-Latch gebildet werden. Das Vorschalten von G_{11} und G_{12} dient hier lediglich der Verwirrung. C_1 muss high sein.
LD_2 entspricht hier unsrem Q, LD_3 unsrem $\overline{Q}$. BTN_1 bewirkt einen reset.


LD7
LD6
LD5
LD4
LD3
LD2
LD1
LD0
SW3
SW2
SW1
SW0
BTN0
BTN1
BTN2
Pin 1
GND
VCCO
VCC
S1
SW0
LD0
R1
SW1
LD1
S2
SW2
LD2
R2
SW3
LD3
BTN2
C2
BTN2
BTN2
C1
BTN2
E3
E2
CLK
E1
CLK
E0
RS-Latch
RS-Latch
G5
G6
G7
G8
G9
G10
G11
G12
G1
G2
G3
G4
D
D
D
CLKGEN
Inverter mit Verzögerung D

2 Entwurf eines digitalen Wecker

2.1 Zustandsdiagramm des Weckers



2.2 vollständige Automatentafel

	$\mathbf{S_1S_2}$	$\mathbf{S_1\overline{S_2}}$	$\mathbf{\overline{S_1}S_2}$	$\mathbf{\overline{S_1S_2}}$
$S_{SetTime}$	S_{Time}	$S_{SetTime}$	S_{Time}	S_{Time}
S_{Time}	S_{Time}	$S_{SetTime}$	$S_{SetAlarm}$	S_{Time}
$S_{SetAlarm}$	S_{Time}	S_{Time}	$S_{SetAlarm}$	S_{Time}

2.3 Minimieren von q'_0 und q'_1

$S_2 \backslash S_1$				
	0_0	1_1	1_5	0_4
	0_2	0_3	0_7	0_6
	0_{10}	0_{11}	-15	-14
	0_8	0_9	-13	-12

q_1 | q_0

Tabelle 1: Minimierung von q'_0

$S_2 \backslash S_1$				
	0_0	0_1	0_5	0_4
	1_2	0_3	0_7	0_6
	1_{10}	0_{11}	-15	-14
	0_8	0_9	-13	-12

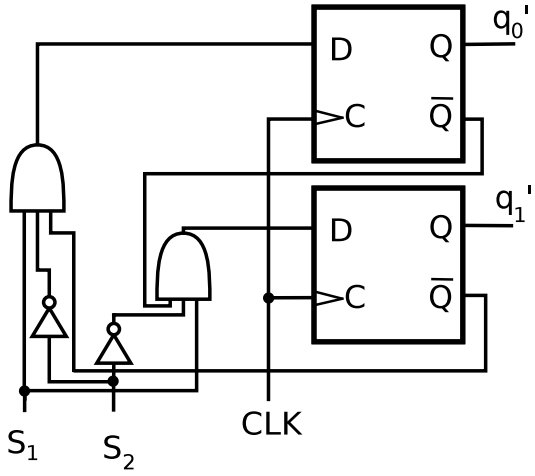
q_1 | q_0

Tabelle 2: Minimierung von q'_1

$$q'_0 = S_1\overline{S_2q_1}$$
$$q'_1 = S_2\overline{S_1q_0}$$

2.4 Schaltwerk aus D-Flipflops

Zum Speichern des aktuellen Zustands werden zwei D-Flipflops verwendet. Zum Berechnen des Folgezustands werden die zuvor ermittelten Formeln eingesetzt.



2.5 Vervollständigen von sevensegment.vhd und controller.vhd

Listing 1: Änderung an der sevensegment.vhd

```
1  [...]
2  -- |-0-|
3  -- 6   1
4  -- |-5-|
5  -- 4   2
6  -- |-3-|
7  CASE NUMBER IS -- Reihenfolge 6543210
8    WHEN 0       => DIGIT_in <= "1011111"
9    WHEN 1       => DIGIT_in <= "0000110"
10   WHEN 2       => DIGIT_in <= "0111011"
11   WHEN 3       => DIGIT_in <= "0101111"
12   WHEN 4       => DIGIT_in <= "1100110"
13   WHEN 5       => DIGIT_in <= "1101101"
14   WHEN 6       => DIGIT_in <= "1111101"
15   WHEN 7       => DIGIT_in <= "0000111"
16   WHEN 8       => DIGIT_in <= "1111111"
17   WHEN 9       => DIGIT_in <= "1101111"
18   WHEN others => DIGIT_in <= "0000000" -- nichts
19 END CASE;
20
21 [...]
```

Listing 2: Änderung an der controller.vhd

```
1  [...]
2
3  CASE current_state IS
4          -- Zustand Time
5          WHEN ntime      =>
6      IF (hours = whours AND mins = wmins AND S(5) = '1') THEN
7        alarm <= '1'; -- ringring, ringring :D
8      ELSE
9        alarm <= '0';
10     END IF;
11           -- Pruefe, ob Alarm ausgeloest werden muss
12
13         -- Zustand SetTime
14         WHEN set_time  =>
15     IF (S(3) = '1' AND sectrigger = '1') THEN -- Minute hochzaehlen
16       IF mins = 59  THEN
17         mins <= 0;
18       ELSE
19         mins <= mins+1;
20       END IF;
21     ELSIF (S(4) = '1' AND sectrigger = '1') THEN -- Stunde hochzaehlen
22       IF hours = 23 THEN
23         hours <= 0;
24       ELSE
25         hours <= hours+1;
26       END IF;
27     END IF
28   -- Zustand SetAlarm
29         WHEN set_alarm =>
30     IF (S(3) = '1' AND sectrigger = '1') THEN -- Minute hochzaehlen
31       IF wmins = 59 THEN
32         wmins <= 0;
33       ELSE
34         wmins <= wmins+1;
35       END IF;
36     ELSIF (S(4) = '1' AND sectrigger = '1') THEN -- Stunde hochzaehlen
37       IF whours = 23 THEN
38         whours <= 0;
39       ELSE
40         whours <= whours+1;
41       END IF;
42     END IF;
43         -- Illegale Zustaende
44         WHEN others     =>
45     -- da sollten wir nie reinkommen
46 END CASE;
47
48 [...]
```