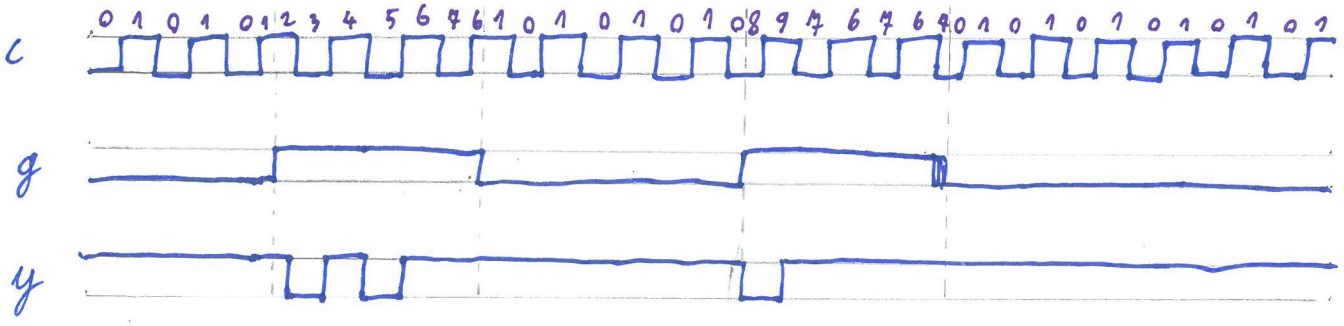
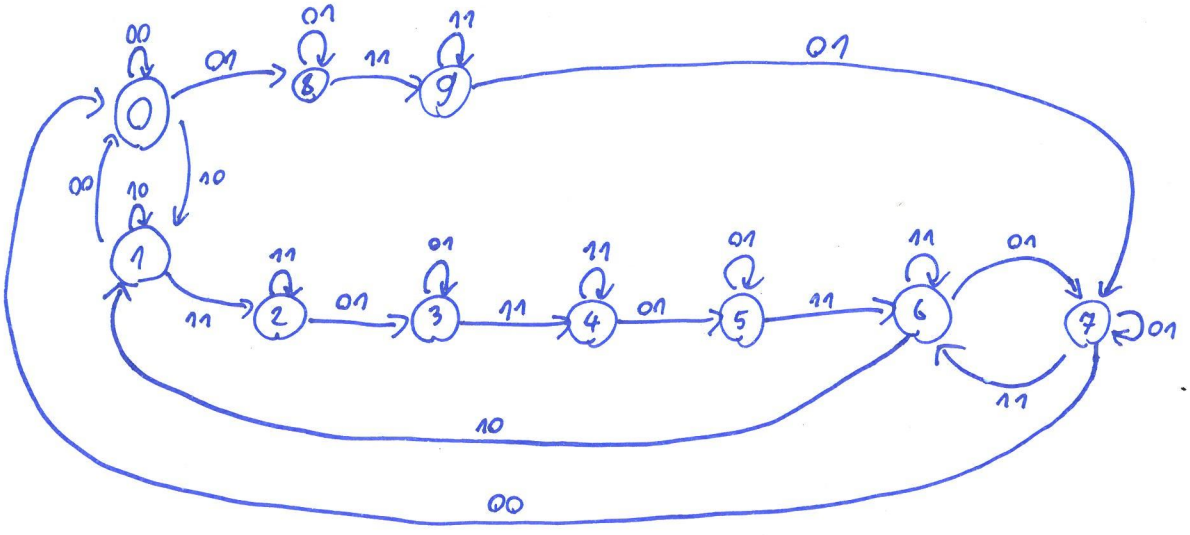
Timing diagram



state	description
0	clock low, gate low
1	clock high, gate low
2	positive slope detected at high clock, waiting for clock change
3	first pulse
4	delay
5	second pulse
6	clock high, gate high
7	clock low, gate high
8	positive slope detected at low clock, waiting for clock change
9	first and only pulse

state transition graph



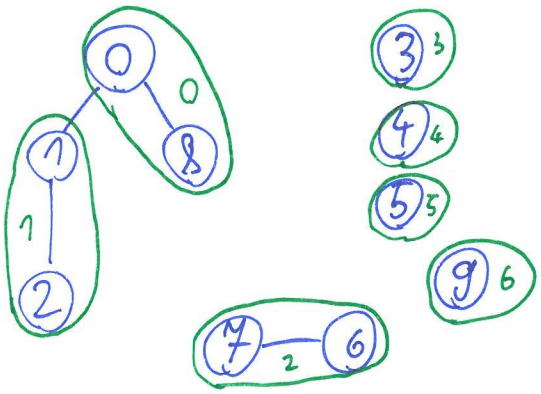
primitive state transition table

s \ cq	00	01	11	10	y
0	(0)	8	-	1	1
1	0	-	2	(1)	1
2	-	3	(2)	-	1
3	-	(3)	4	-	0
4	-	5	(4)	-	1
5	-	(5)	6	-	0
6	-	7	(6)	1	1
7	0	(7)	6	-	1
8	-	(8)	9	-	1
9	-	7	(9)	-	0

implication matrix

	0	1	2	3	4	5	6	7	8
1	J								
2	3,8	J							
3	X	X	X						
4	5,8	2,4	3,5	X					
5	X	X	X	4,6	X				
6	7,8	2,6	3,7	X	5,7	X			
7	7,8	2,6	3,4 2,6	X	5,7	X	J		
8	J	2,9	3,6 2,9	X	5,8 4,9	X	7,8 6,9	6,9	
9	X	X	X	3,4 4,9	X	5,7 6,9	X	X	X

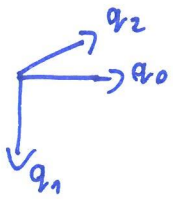
equivalence graph

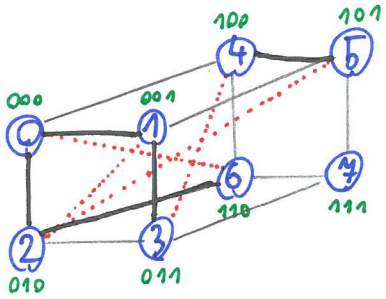


minimized state transition table

s \ cq	00	01	11	10	y
0	(0)	(0)	6	1	1
1	0	3	(1)	(1)	1
2	0	(2)	(2)	1	1
3	-	(3)	4	-	0
4	-	5	(4)	-	1
5	-	(5)	2	-	0
6	-	2	(6)	-	0

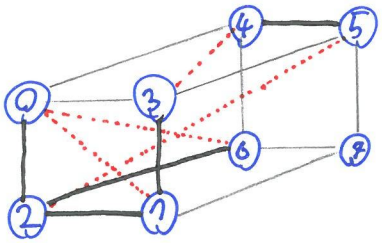
hypercube state code assignment



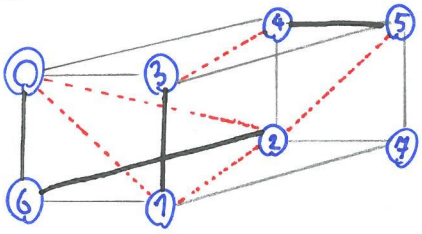


s \ cy	00	01	11	10	y
0	0	0	6	4	1
1	0	3	3	0	1
2	0	2	2	1	1
3	–	3	4	–	0
4	–	5	4	–	1
5	–	5	2	–	0
6	–	2	6	–	0

swap 1 and 3

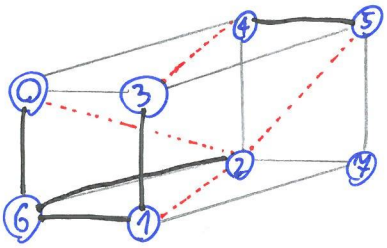


swap 2 and 6

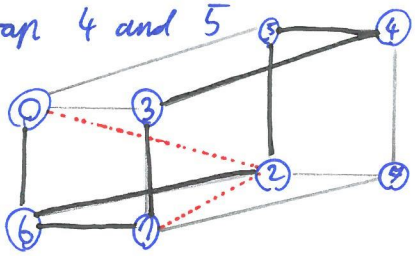


reroute 0→1 and 1→0 through 6

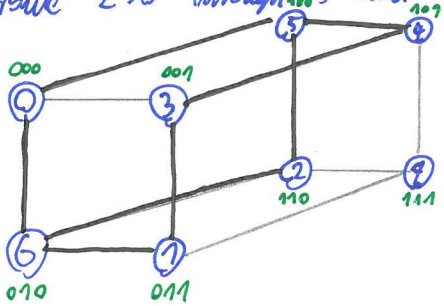
s \ cy	00	01	11	10	y
0	0	0	6	6	1
1	6	3	3	1	1
2	0	2	2	1	1
3	–	3	4	–	0
4	–	5	4	–	1
5	–	5	2	–	0
6	0	2	6	1	0



swap 4 and 5



reroute 2→0 through 5 and 2→1 through 6



s \ cy	00	01	11	10	y
0	0	0	6	6	1
1	6	3	3	3	1
2	5	2	2	4	1
3	–	3	4	–	0
4	–	5	4	–	1
5	0	5	2	–	0
6	0	2	6	1	0

final state code assignment

state	q_2	q_1	q_0	y
0	0	0	0	1
1	0	1	1	1
2	1	1	0	1
3	0	0	1	0
4	1	0	1	1
5	1	0	0	0
6	0	1	0	0

final state transition table

s \ cg	00	01	11	10	y
0	(0)	(0)	6	6	1
1	6	3	(1)	(1)	1
2	5	(2)	(2)	6	1
3	–	(3)	4	–	0
4	–	5	(4)	–	1
5	0	(5)	2	–	0
6	(1)	2	(6)	1	0

SR latch behavior

S	R	result
0	0	no change
0	1	0
1	0	1
1	1	illegal

SR latch application table

Q → Q'		S	R
0	0	0	–
0	1	1	0
1	0	0	1
1	1	–	0

state transition table with SR input

state	code	input cg	next state	code	q_2 S_2	q_2 R_2	q_1 S_1	q_1 R_1	q_0 S_0	q_0 R_0
0	000	00	0	000	0	–	0	–	0	–
0	000	01	0	000	0	–	0	–	0	–
0	000	11	6	010	0	–	1	0	0	–
0	000	10	6	010	0	–	1	0	0	–
1	011	00	6	010	0	–	–	0	0	1
1	011	01	3	001	0	–	0	1	–	0
1	011	11	1	011	0	–	–	0	–	0
1	011	10	1	011	0	–	–	0	–	0
2	110	00	5	100	–	0	0	1	0	–
2	110	01	2	110	–	0	–	0	0	–
2	110	11	2	110	–	0	–	0	0	–
2	110	10	6	010	0	1	–	0	0	–
3	001	01	3	001	0	–	0	–	–	0
3	001	11	4	101	1	0	0	–	–	0
4	101	01	5	100	–	0	0	–	0	1
4	101	11	4	101	–	0	0	–	–	0
5	100	00	0	000	0	1	0	–	0	–
5	100	01	5	100	–	0	0	–	0	–
5	100	11	2	110	–	0	1	0	0	–
6	010	00	0	000	0	–	0	1	0	–
6	010	01	2	110	1	0	–	0	0	–
6	010	11	6	010	0	–	–	0	0	–
6	010	10	1	011	0	–	–	0	1	0

Karnaugh maps for SR inputs

R_0

$q_2q_1q_0$ \ cq	00	01	11	10
000	–	–	–	–
001	–	0	0	–
011	1	0	0	0
010	–	–	–	0
110	–	–	–	–
111	–	–	–	–
101	–	1	0	–
100	–	–	–	–

$R_0 = \bar{c}\bar{q} + \bar{c}q_2$

S_0

$q_2q_1q_0$ \ cq	00	01	11	10
000	0	0	0	0
001	–	–	–	–
011	0	–	–	–
010	0	0	0	1
110	0	0	0	0
111	–	–	–	–
101	–	0	–	–
100	0	0	0	–

$S_0 = c\bar{q}\,\bar{q_2}\,q_1$

R_1

$q_2q_1q_0$ \ cq	00	01	11	10
000	1	–	0	0
001	–	1	–	–
011	0	1	0	0
010	1	0	0	0
110	1	0	0	0
111	–	1	–	–
101	–	0	–	–
100	0	–	0	–

$R_1 = (\bar{q} \oplus q_0)\bar{c}$

S_1

$q_2q_1q_0$ \ cq	00	01	11	10
000	0	0	1	1
001	–	0	0	–
011	–	0	–	–
010	0	–	–	–
110	0	–	–	–
111	–	–	–	–
101	–	0	0	–
100	0	0	1	–

$S_1 = c\,\bar{q_0}$

R_2

$q_2q_1q_0$ \ cq	00	01	11	10
000	–	–	–	–
001	–	–	0	–
011	–	–	–	–
010	–	0	–	–
110	0	0	0	1
111	–	–	–	–
101	–	0	0	–
100	1	0	0	–

$R_2 = c\bar{q} + \bar{q}\,\bar{q_1}$

S_2

$q_2q_1q_0$ \ cq	00	01	11	10
000	0	0	0	0
001	–	0	1	–
011	0	0	0	0
010	0	1	0	0
110	–	–	–	0
111	–	–	–	–
101	–	–	–	–
100	0	–	–	–

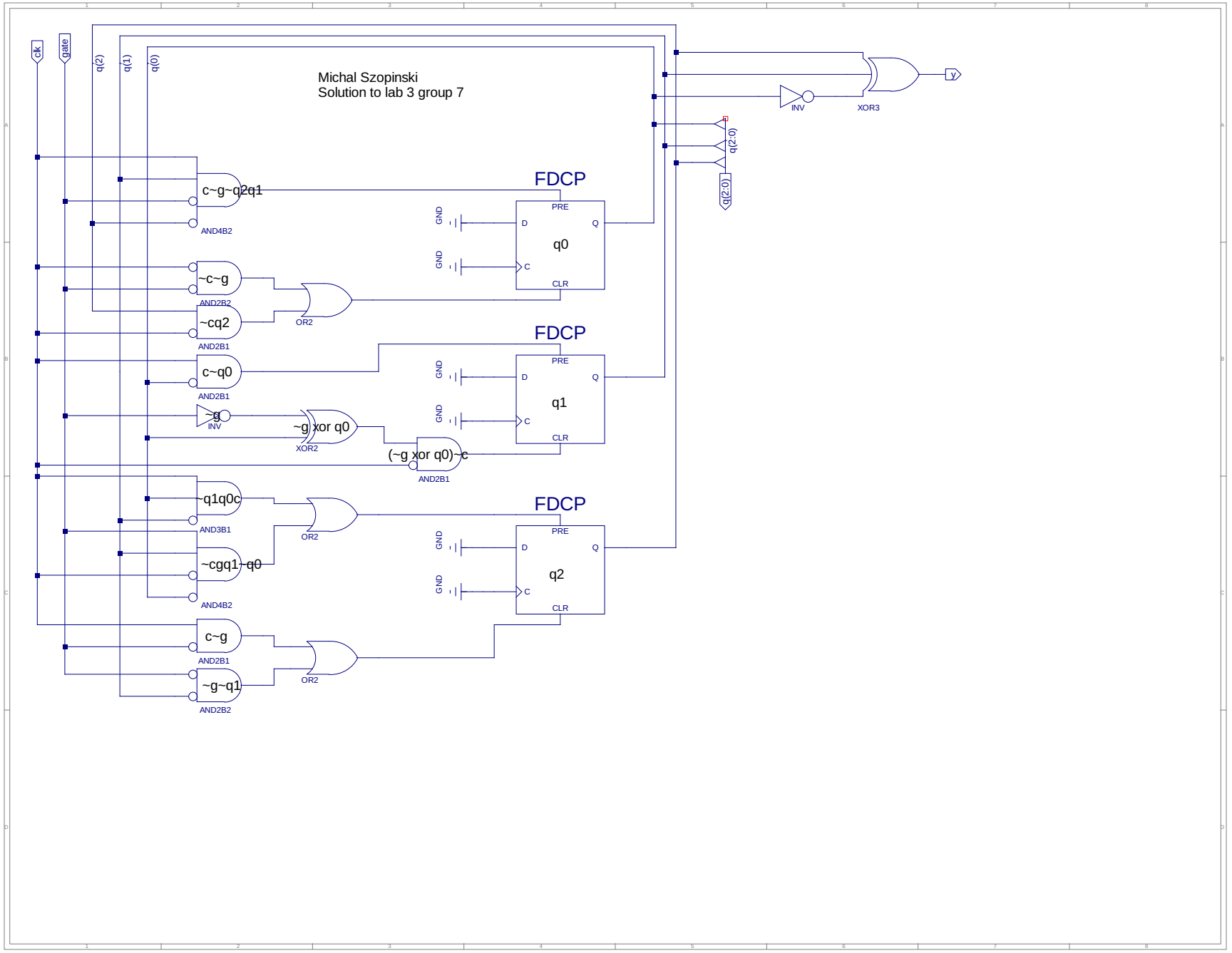
$S_2 = \bar{q_1}\,q_0\,c + \bar{c}q\,q_1\bar{q_0}$

Karnaugh map for output

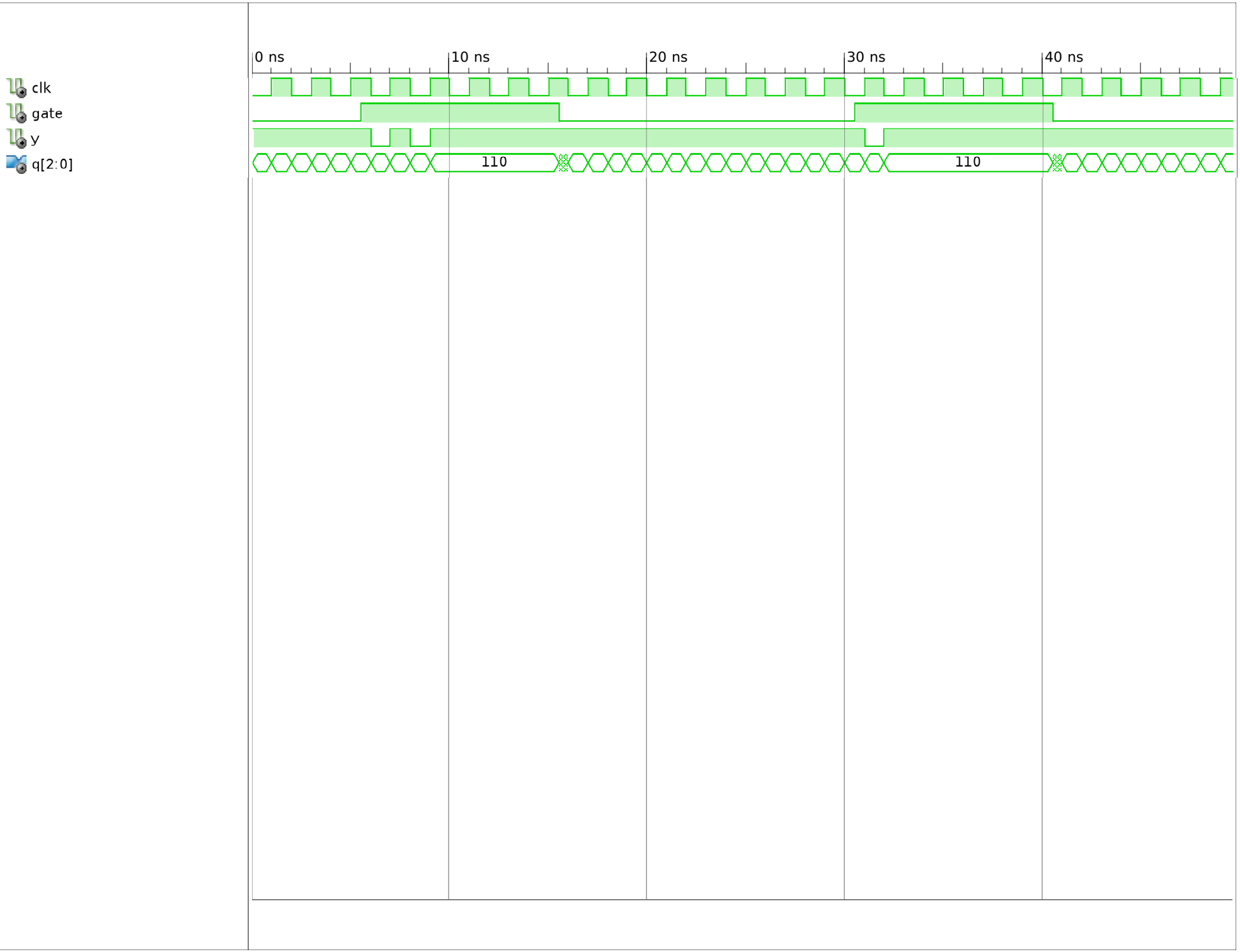
y

q_2q_1 \ q_0	0	1
00	1	0
01	0	1
11	1	–
10	0	1

$y = q_2 \oplus q_1 \oplus \bar{q_0}$


Michal Szopinski
Solution to lab 3 group 7
clk
gate
q(2)
q(1)
q(0)
c~g~q2q1
AND4B2
~c~g
AND2B2
~cq2
AND2B1
OR2
FDCP
PRE
D
Q
q0
C
CLR
GND
c~q0
AND2B1
~g
INV
~g xor q0
XOR2
(~g xor q0)~c
AND2B1
FDCP
q1
~q1q0c
AND3B1
~cgq1~q0
AND4B2
OR2
FDCP
q2
c~g
AND2B1
~g~q1
AND2B2
OR2
INV
XOR3
y
q(2:0)
q(2:0)

```vhdl
-- Vhdl test bench created from schematic /home/mszopinski/Desktop/labko3/schem.sch - Sat Dec  7 17:17:06 2019
--
-- Notes:
-- 1) This testbench template has been automatically generated using types
-- std_logic and std_logic_vector for the ports of the unit under test.
-- Xilinx recommends that these types always be used for the top-level
-- I/O of a design in order to guarantee that the testbench will bind
-- correctly to the timing (post-route) simulation model.
-- 2) To use this template as your testbench, change the filename to any
-- name of your choice with the extension .vhd, and use the "Source->Add"
-- menu in Project Navigator to import the testbench. Then
-- edit the user defined section below, adding code to generate the
-- stimulus for your design.
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY schem_schem_sch_tb IS
END schem_schem_sch_tb;
ARCHITECTURE behavioral OF schem_schem_sch_tb IS

   COMPONENT schem
   PORT( clk    :   IN  STD_LOGIC;
         gate   :   IN  STD_LOGIC;
         y      :   OUT STD_LOGIC;
         q      :   OUT STD_LOGIC_VECTOR (2 DOWNTO 0));
   END COMPONENT;

   SIGNAL clk   :   STD_LOGIC := '0'; -- SET DEFAULT VALUES
   SIGNAL gate  :   STD_LOGIC := '0';
   SIGNAL y     :   STD_LOGIC;
   SIGNAL q     :   STD_LOGIC_VECTOR (2 DOWNTO 0);
BEGIN

   UUT: schem PORT MAP(
        clk => clk,
        gate => gate,
        y => y,
        q => q
   );

    -- SET CLOCK
    clk <= not clk after 1 ns;

-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
        wait for 5.5 ns;
        gate <= '1';
        wait for 10 ns;
        gate <= '0';

        wait for 15 ns;
        gate <= '1';
        wait for 10 ns;
        gate <= '0';

      WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;

```


0 ns
10 ns
20 ns
30 ns
40 ns
clk
gate
y
q[2:0]
110
110