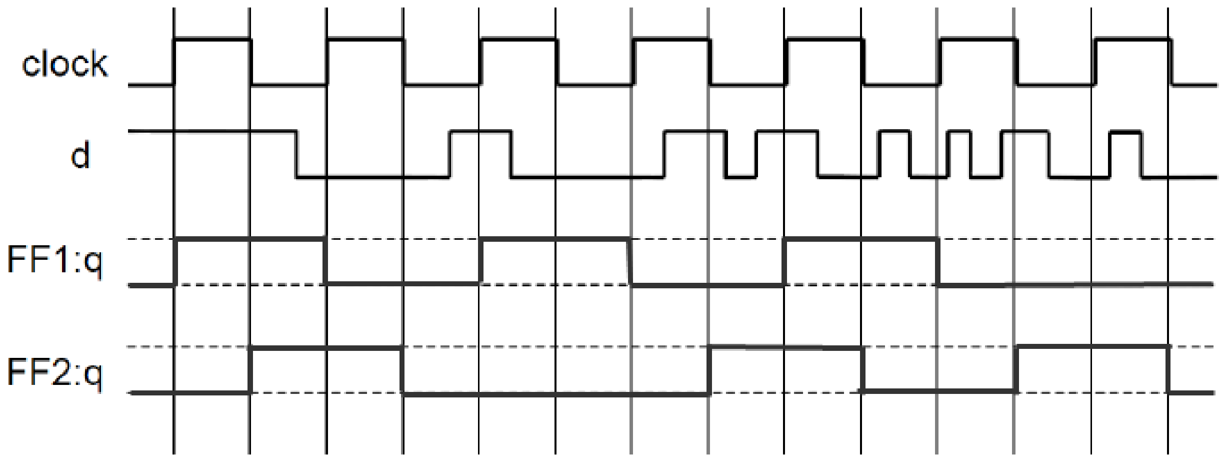
Technische Informatik: Abgabe 7

Michael Mardaus Andrey Tyukin

12. Dezember 2013

Exercise 7.1 (Flipflop logical simulation)

Let FF1 be a $0 \to 1$ flank controlled D-flipflop and FF2 a $1 \to 0$ flank controlled one.



Exercise 7.2 (Flipflops)

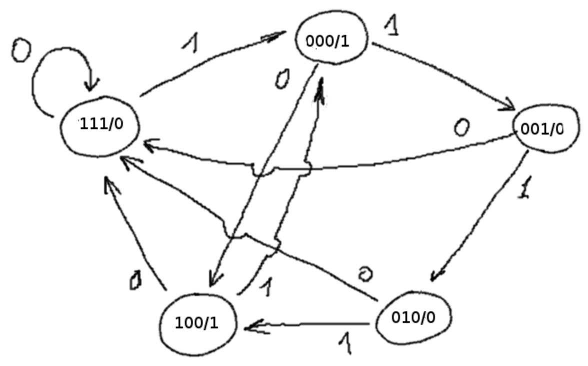
a) The state diagram for the automaton:

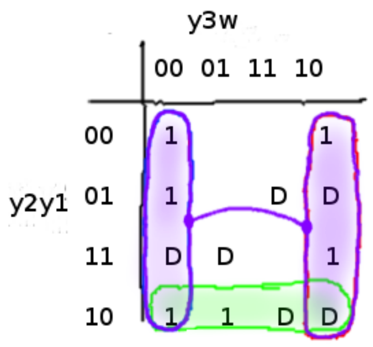
b) For the implementation with D-Flip-Flops, we construct the K-Maps first (notice that there are quite a few Don't-Care entries):

Possible (nearly) optimal formulas are:

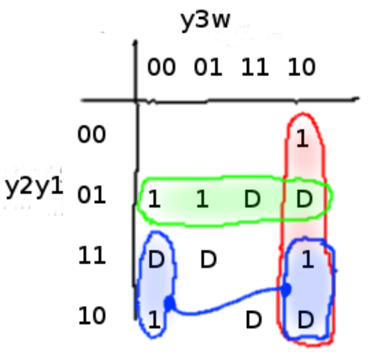
$$Y_3 = \bar{w} + y_2\bar{y}_1$$
$$Y_2 = \bar{y}_3 y_1 + y_2\bar{w} + y_3\bar{w}$$
$$Y_1 = y_1\bar{w} + y_2\bar{w} + y_3\bar{w} + \bar{y}_1\bar{y}_2\bar{y}_3 w$$

This gives us the following circuit:

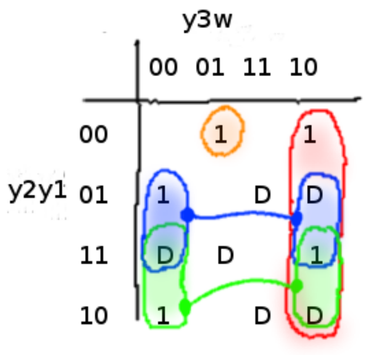




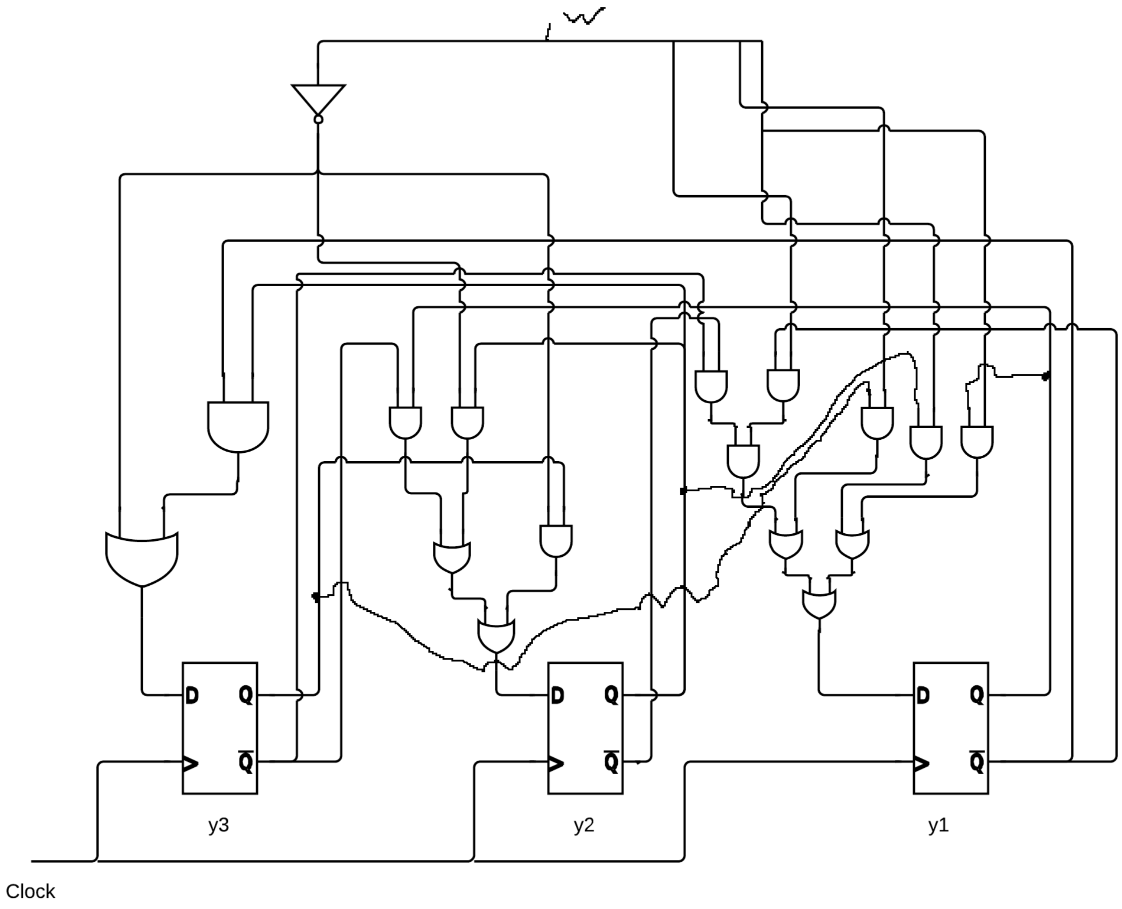
y2y1 \ y3w	00	01	11	10
00	1			1
01	1		D	D
11	D	D		1
10	1	1	D	D



y2y1 \ y3w	00	01	11	10
00				1
01	1	1	D	D
11	D	D		1
10	1		D	D



y2y1 \ y3w	00	01	11	10
00		1		1
01	1		D	D
11	D	D		1
10	1		D	D


D
Q
Q̄
y3
y2
y1
Clock

c) (omitted)

Exercise 7.3 (Gum machine)

We contructed an automaton with 4 states, as we do not differentiate between 15 cents and 20 cents (and even more cents). (That way we have smaller tables.)

- state 00 means 0 cents balance
- state 01 means 10 cents balance
- state 10 means 5 cents balance
- state 11 means „enough cents“ balance (≥ 15)

We have one input bit w, which is 0 if a 5 cent coin is insered and 1 if a 10 cent coin is inserted. That gives us the following state/output table:

State	Next state		Output	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
y_1y_0	Y_1Y_0	Y_1Y_0	z	z
00	10	01	0	0
01	11	11	0	0
10	01	11	0	0
11	11	11	1	1

We extract the K-maps for Y_0 and Y_1 and z and get

Y_0	y_1y_0			
w	00	01	11	10
0		1	1	1
1	1	1	1	1

$Y_0 = w + y_0 + y_1$

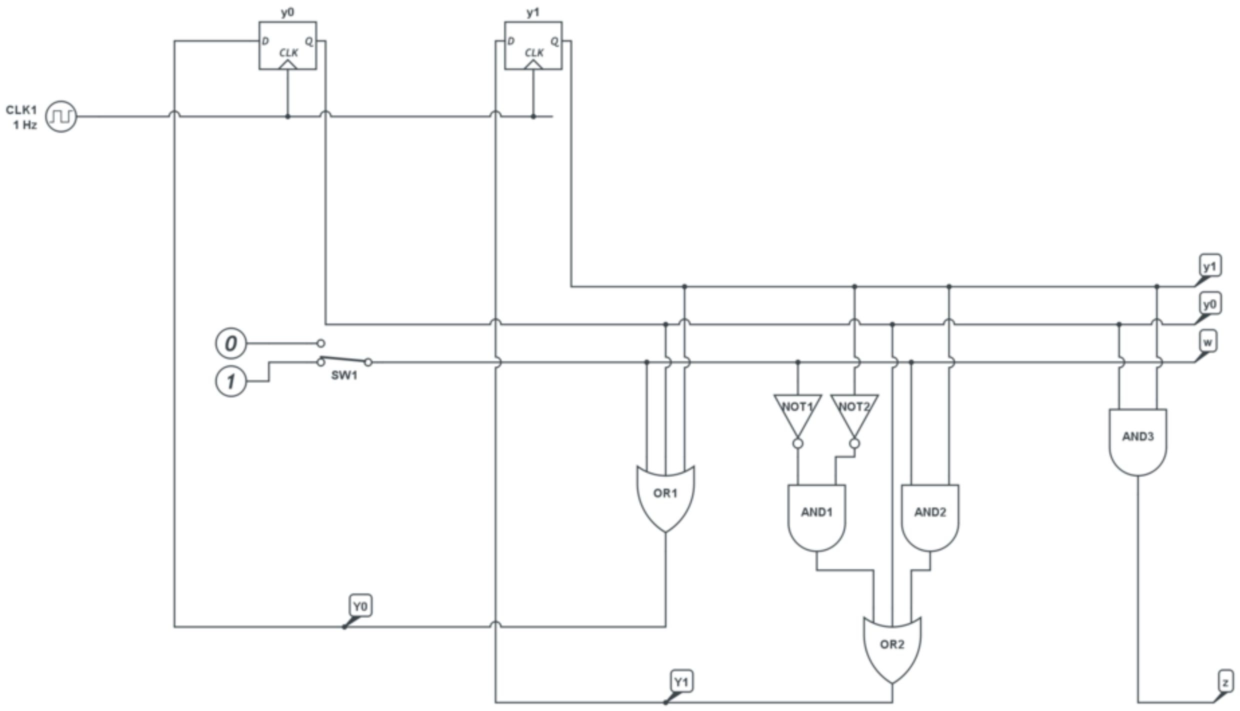
Y_1	y_1y_0			
w	00	01	11	10
0	1	1	1	
1		1	1	1

$Y_1 = \bar{w}\bar{y}_1 + y_0 + wy_1$

z	y_1y_0			
w	00	01	11	10
0			1	
1			1	

$z = y_1y_2$

and that leads us to this circuit:



Exercise 7.4 (von Neumann-Adder)

Let A be a von Neumann adder with registers of size n. We want to describe this adder as a finite automaton. For this, we let $Q := \mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B} \times \mathbb{B}$ be the set of possible states, and interpret it as follows: accumulator $\in \mathbb{B}^n$, carry $\in \mathbb{B}^n$, highOrderBit $\in \mathbb{B}$, moreWorkToDo $\in \mathbb{B}$ (in this order). As in the lecture and the book, we are not concerned with reading the inputs into the registers or writing the results to somewhere else, so we set input/output alphabets $\Sigma, \Delta = \emptyset$ and let the start state q_0 be arbitrary from $\mathbb{B}^n \times \mathbb{B}^n \times \{0\} \times \{1\}$ (in the beginning, the highest order carry-bit should not be set, and the moreWorkToDo shall indicate that there is something to do). The end states can be set to $F = \mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B} \times \{0\}$. The transition function is then as follows (colon means that we prepend a zero to the tuple, empty word ϵ is omitted on both sides):

$$\delta\left((a_i)_{i=1}^n, (c_i)_{i=1}^n, h, t\right) = \left((a_i \not\leftrightarrow b_i)_{i=1}^n, 0 : (a_{i-1} \wedge b_{i-1})_{i=2}^n, h \vee (a_n \wedge b_n), \bigvee_{i=2}^n (a_{i-1} \wedge b_{i-1})\right).$$

Just to verify that this indeed does what we want, we implemented it literally, here is an example of adding $568468843 + 708052434$ in 30-bit registers:

```
Acc:   110101101010010001000111100001
Carry: 010010111010000000101100010101
Highest bit: false
Acc:   100111010000010001101011110100
Carry: 001000010101000000000010000000
Highest bit: true
Acc:   101111000101010001101001110100
Carry: 000000001000000000000001000000
Highest bit: true
Acc:   101111001101010001101000110100
Carry: 000000000000000000000000100000
```

```
Highest bit: true
Acc:   1011110011010100011010000010100
Carry: 0000000000000000000000000010000
Highest bit: true
Acc:   1011110011010100011010000000100
Carry: 0000000000000000000000000001000
Highest bit: true
1276521277

$ echo $((568468843+708052434))
1276521277
```