School of Engineering
InES Institute of Embedded Systems

Combinational and Sequential Logic

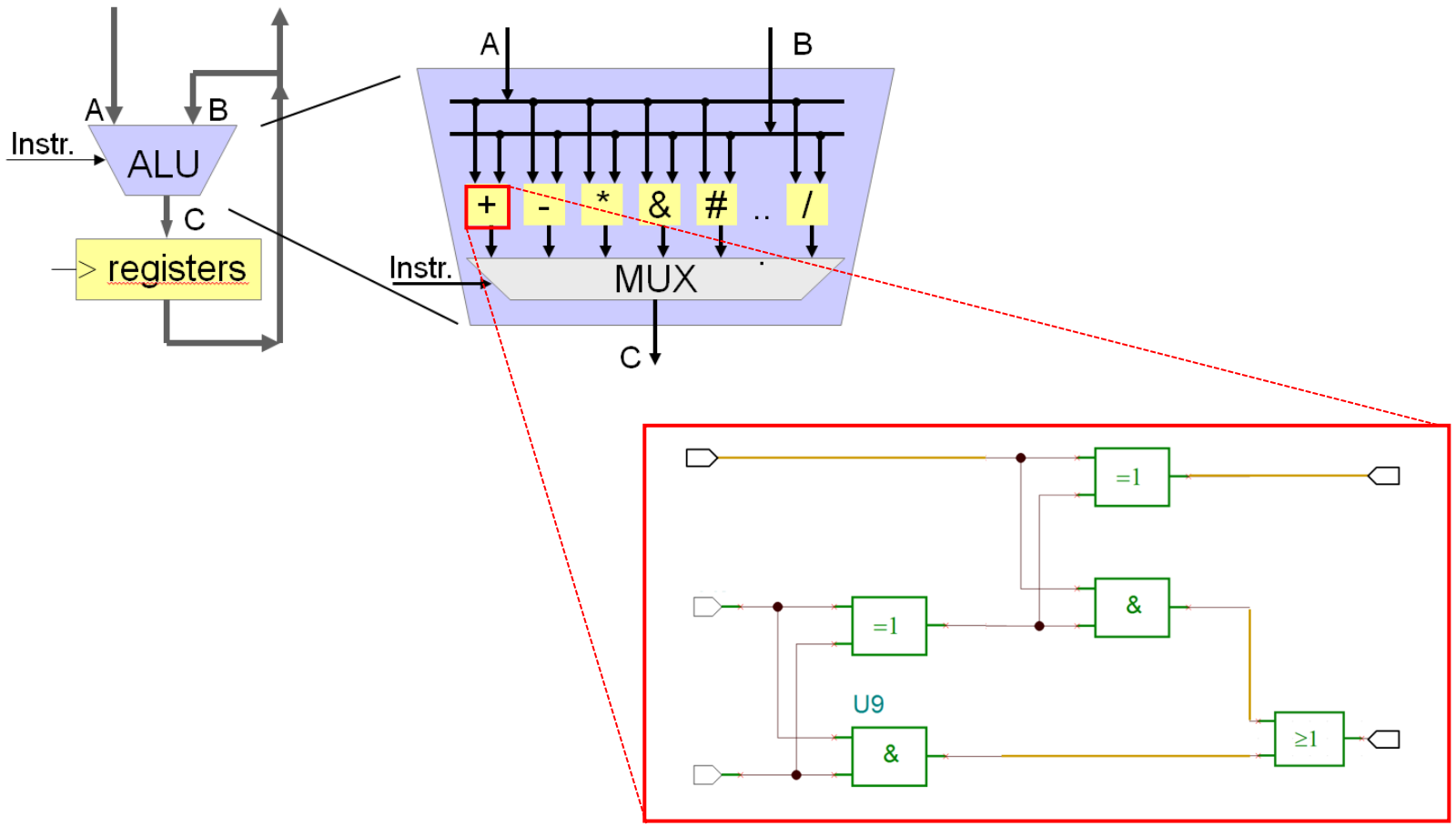
Computer Engineering 1

CT Team: A. Gieriet, J. Gruber, R. Gübeli, M. Meli, M. Rosenthal, A. Rüst, J. Scheier, M. Thaler

animation

Motivation

- **Arithmetical and Logical Unit (ALU)**



1 - Bit Full Adder (FA)

Motivation

- **Example:**
  - 16-bit Processor built from transistors

http://www.megaprocessor.com

Motivation

School of Engineering
InES Institute of Embedded Systems

Combinational Logic

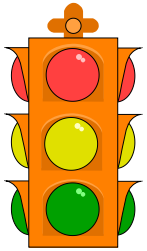
Leap year if

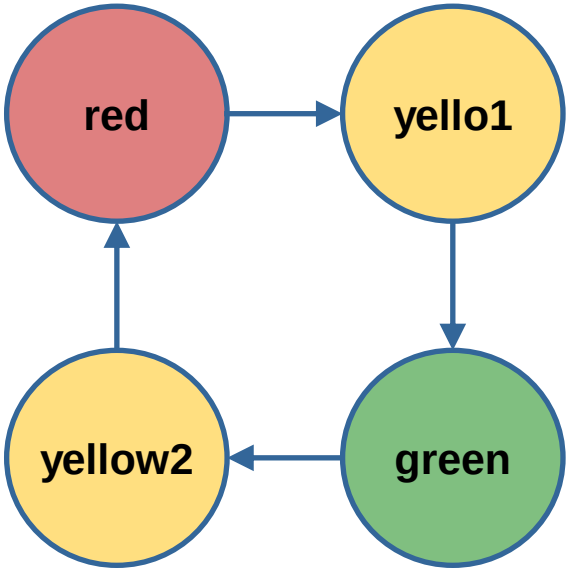
year divisible by 4

AND

year **NOT** divisible by 100
OR
year divisible by 400

Sequential Logic





Learning Objectives

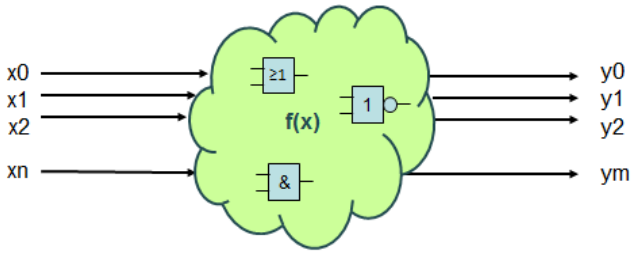
School of Engineering
InES Institute of Embedded Systems

- **At the end of this lesson you will be able**
  - to explain what combinational logic is and to enumerate the basic logic operations
  - to interpret a schematic of a simple combinational circuit and to derive the associated truth table
  - to differentiate between combinational and sequential logic
  - to explain the function of a D-Flip-Flop from a user's perspective
  - to apply the relation between frequency and period for clock signals including associated units and dimensions
  - to interpret simple sequential circuits and to derive associated truth tables, timing diagrams and state diagrams
  - to classy circuits as counters, shift register or Moore machine (finite state machine)
  - to explain the function and use of registers and shift registers

Objectives

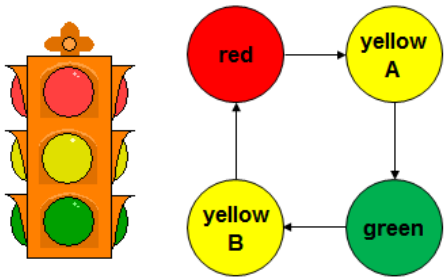
Combinational Logic

- **Basic Logic Operations (Short Repetition INCO)**
  - Symbols / Logic equations / Truth tables
- **Examples**
  - Multiplexer
  - Half-Adder
  - Full-Adder



Sequential Logic

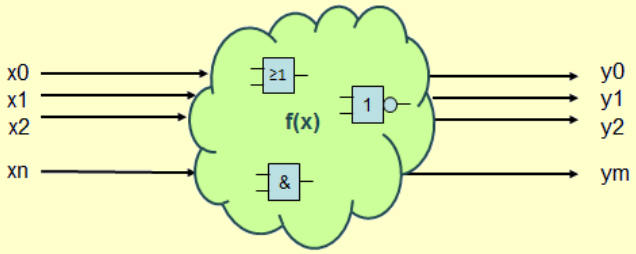
- **Clock Signal**
- **D-Flip-Flop**
- **Timing Diagram**
- **Counter**
  - Example Traffic Light
  - Exercises
- **Register / Shift Register**



Agenda

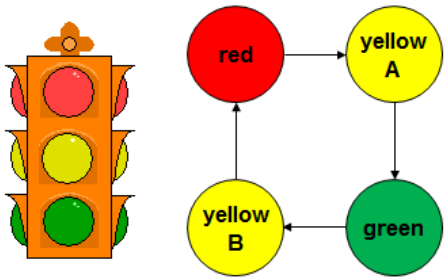
Combinational Logic

- **Basic Logic Operations (Short Repetition INCO)**
  - Symbols / Logic equations / Truth tables
- **Examples**
  - Multiplexer
  - Half-Adder
  - Full-Adder



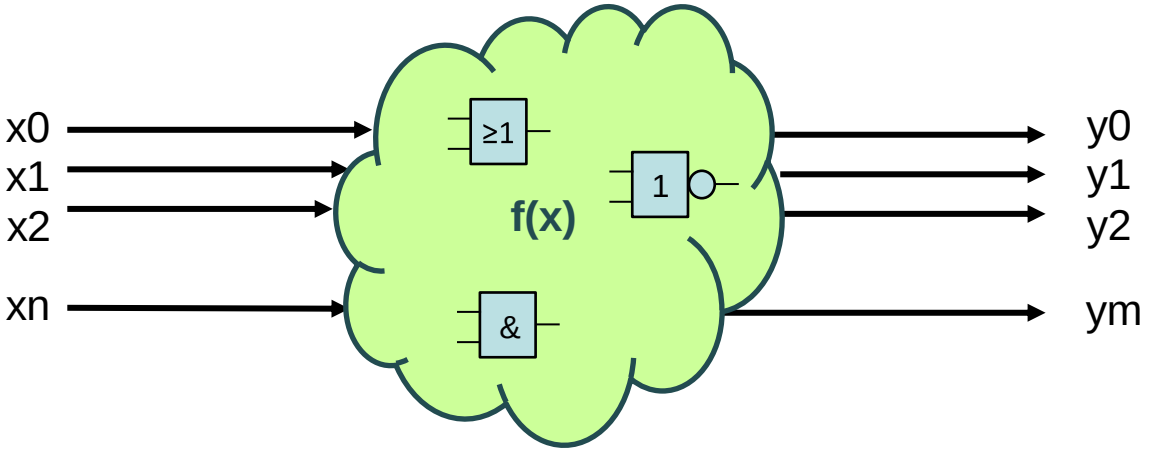
Sequential Logic

- **Clock Signal**
- **D-Flip-Flop**
- **Timing Diagram**
- **Counter**
  - Example Traffic Light
  - Exercises
- **Register / Shift Register**



Combinational Logic

- **Logic states in a binary system**
  - Outputs change depending on inputs and internal logic functions
  - System has no memory, i.e. there is no storage element
  - For N inputs there are 2^N possible input combinations
  - The only timing influence are internal delays
  - Outputs are stable after a delay



Combinational Logic

Basic Logic Operations

Inverter

symbol

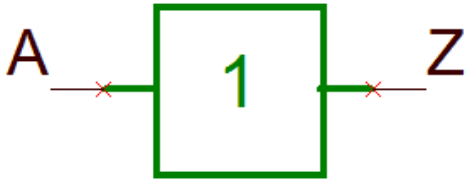
logic equation

Z = !A

truth table

A	Z
0	1
1	0

Buffer

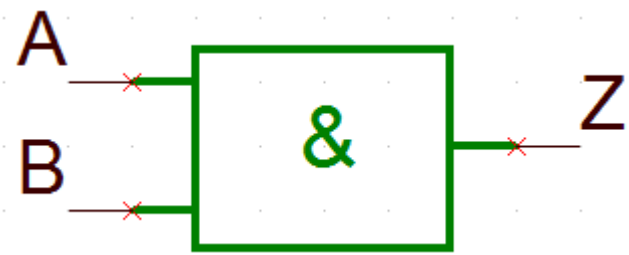


Z = A

A	Z
0	0
1	1

Basic Logic Operations

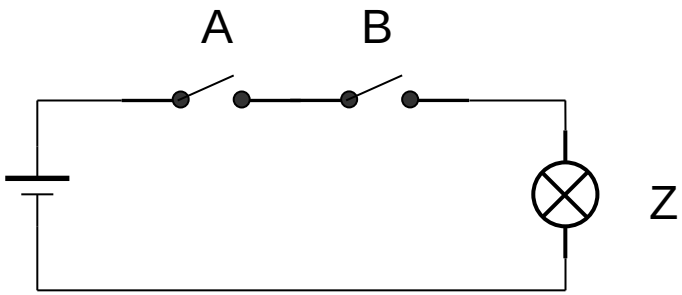
AND



$Z = A \& B$

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

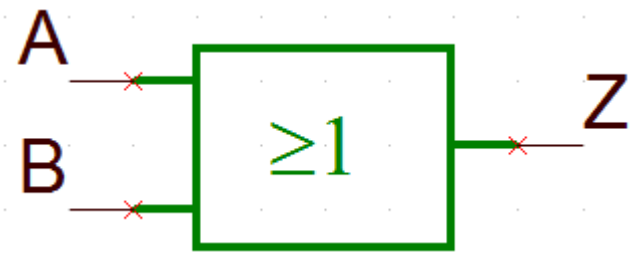
1: switch closed
0: switch open



1: lamp on
0: lamp off

Basic Logic Operations

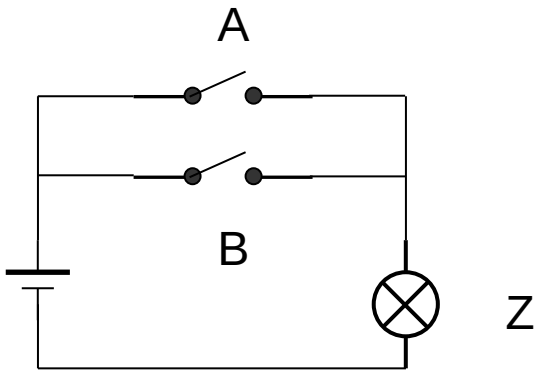
OR



$Z = A \# B$

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

1: switch closed
0: switch open



1: lamp on
0: lamp off

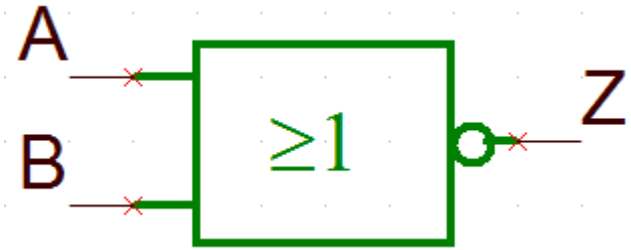
Basic Logic Operations

NAND → NOT AND = AND with inverter

Z = !(A & B)

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR → NOT OR = OR with inverter

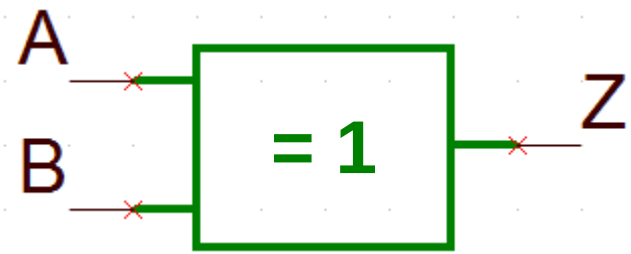


Z = !(A # B)

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

Basic Logic Operations

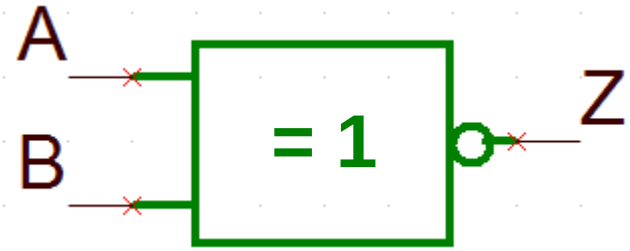
EXOR



Z = A $ B

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

EXNOR



Z = !(A $ B)

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

Symbol Sets

Function	IEC 60617-12 *since 1997*	US ANSI 91 *1984*	DIN 40700 *until 1976*
AND	&		
OR	≥ 1		
Buffer	1		
XOR	= 1		
NOT	1		
NAND	&		
NOR	≥ 1		
XNOR	= 1		

Example: Leap Year

School of Engineering
InES Institute of Embedded Systems

- **Leap year if**

year divisible by 4	**AND**	year **NOT** divisible by 100 **OR** year divisible by 400

- **3 inputs**
  - year divisible by 4 → true or false
  - year divisible by 100 → true or false
  - year divisible by 400 → true or false
- **Output**
  - Is a leap year → true or false

Example: Leap Year

- **Draw a digital logic circuit for evaluating a leap year**
  - Inputs:

    A year is divisible by 4

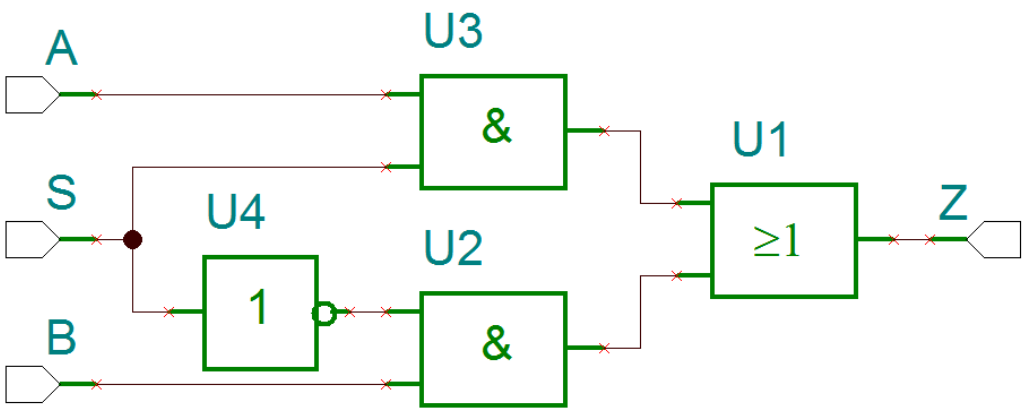
    B year is divisible by 100

    C year is divisible by 400
  - Output:

    Z year is a leap year

- **Fill in the truth table for inputs and output**

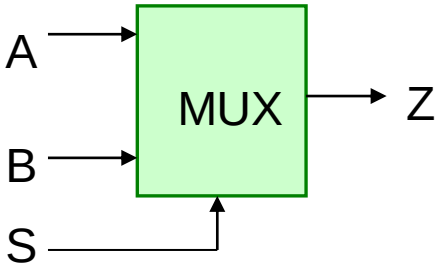
Example: Multiplexer

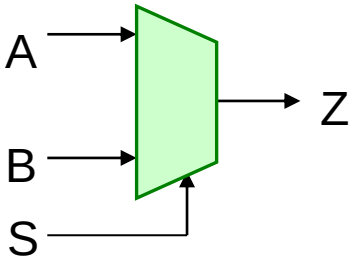


S	A	B	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Z = ?

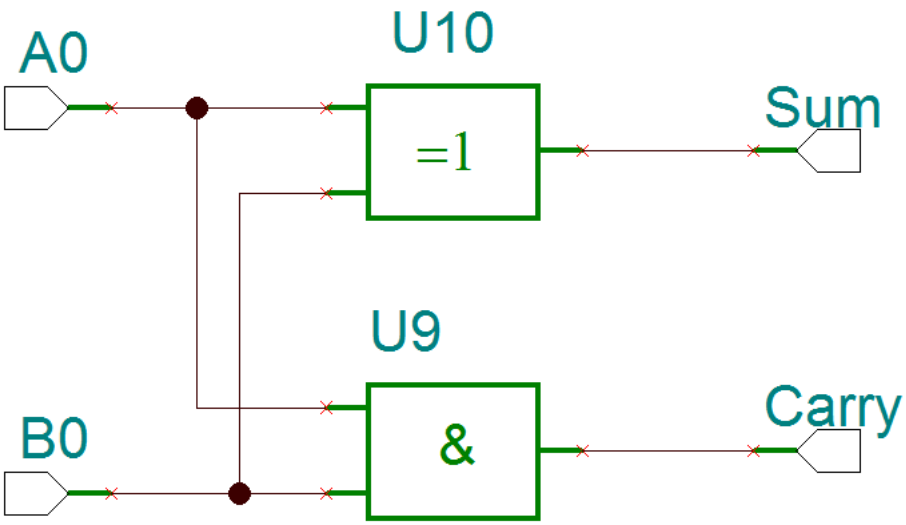
Tasks
- Fill in the truth table
- Compile the logic equation
- What is the function of the circuit?





1-Bit Half-Adder

- **Addition of two 1-Bit inputs**



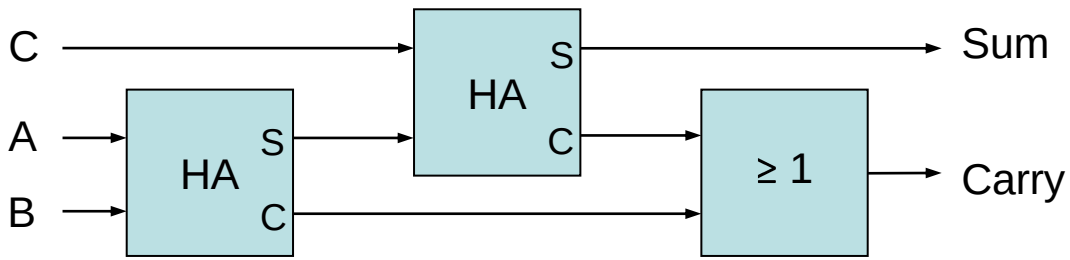
A0	B0	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Sum = A0 $ B0
Carry = A0 & B0

```
    0  1  1  1
+   0  1  0  1
0   1  1  1      Carry
-----------------------
0   1  1  0  0   Sum
```

1-Bit Full-Adder

- **Addition with Carry In**



C	A	B	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

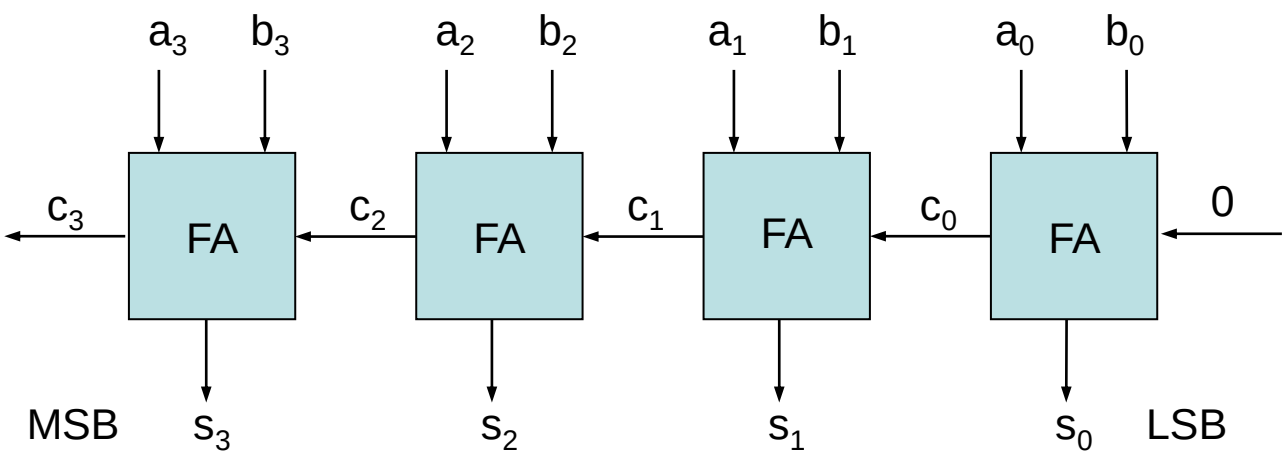
```
    0   1   1   1
+   0   1   0   1
0   1   1   1       Carry
-----------------------
0   1   1   0   0   Sum
```

HA = Half-Adder

4-Bit Adder

```
    0  1  1  1
+   0  1  0  1
0   1  1  1      Carry
-----------------------
0   1  1  0  0   Sum
```

C_{i-1}	A_i	B_i	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



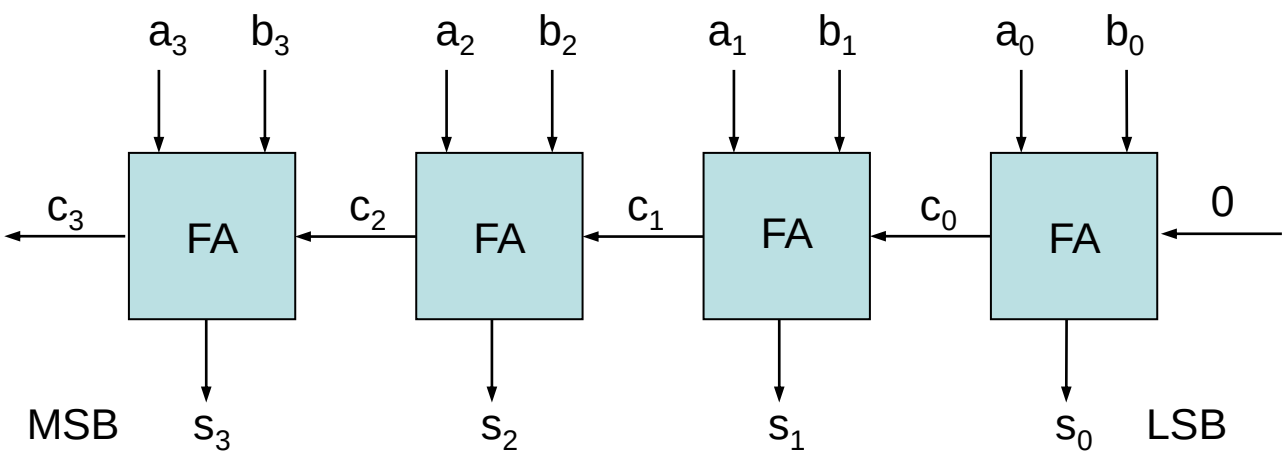
FA = Full-Adder

4-Bit Adder

School of Engineering
InES Institute of Embedded Systems

- **Exercise**
  - Given: $a_3 – a_0 = 13d$ and $b_3 – b_0 = 6d$
  - Convert values to binary and enter them in the figure
  - Use the table to calculate values $s_3 – s_0$ and $c_3 – c_0$ in the figure
  - Check your result by a written binary addition

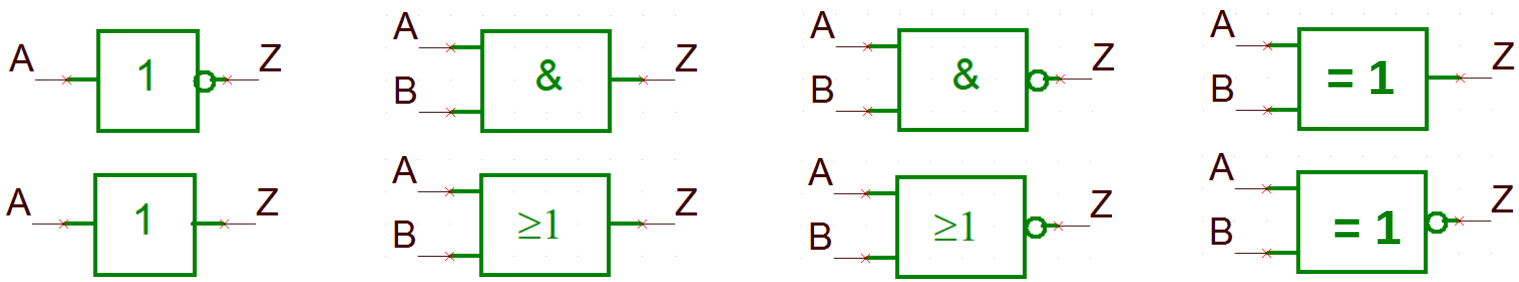
Ci	Ai	Bi	Si	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



FA = Full-Adder

Summary Combinational logic

- **Combinational logic**
  - System <u>without</u> memory (no storage element)
- **Basic logic operations (gates)**

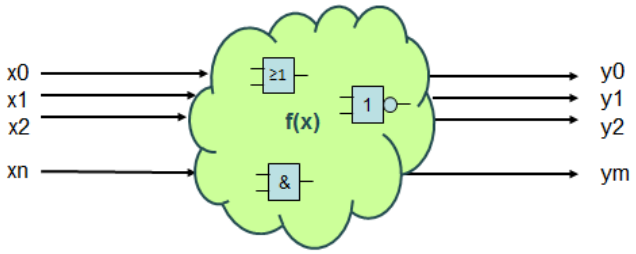


- **Basic circuits**
  - Multiplexer
  - Half adder
  - Full adder

Agenda

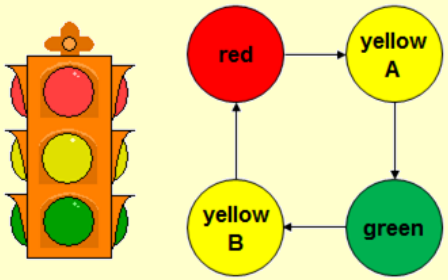
Combinational Logic

- **Basic Logic Operations (Short Repetition INCO)**
  - Symbols / Logic equations / Truth tables
- **Examples**
  - Multiplexer
  - Half-Adder
  - Full-Adder



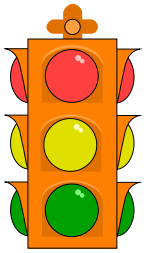
Sequential Logic

- **Clock Signal**
- **D-Flip-Flop**
- **Timing Diagram**
- **Counter**
  - Example Traffic Light
  - Exercises
- **Register / Shift Register**

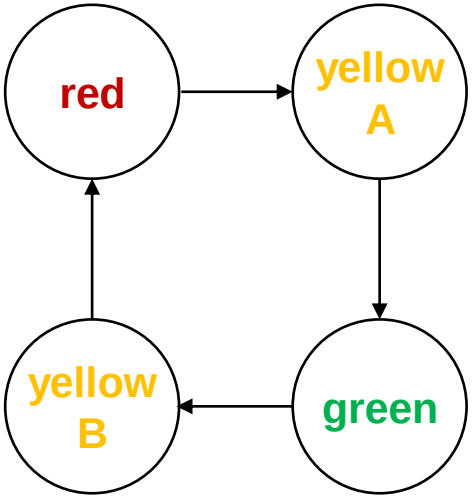


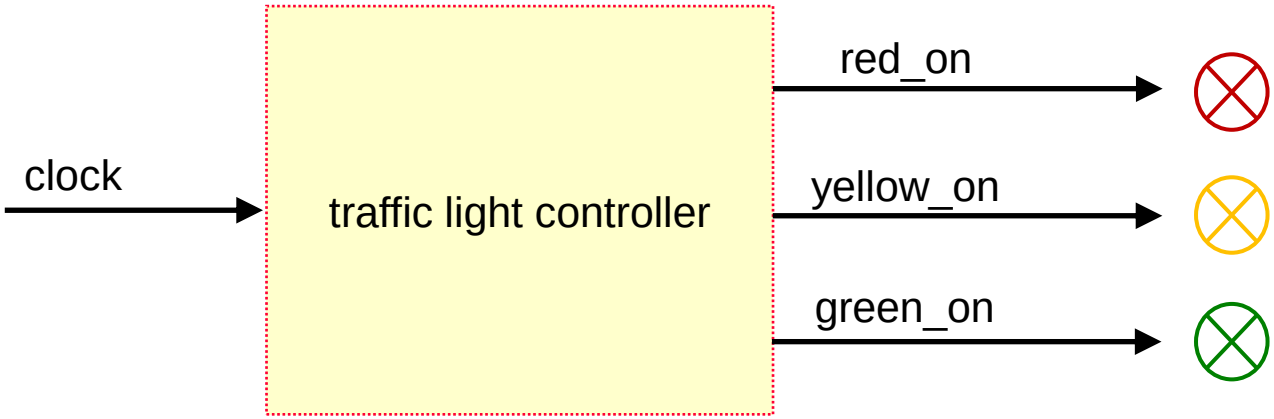
Sequential Logic

- **Example: traffic light**



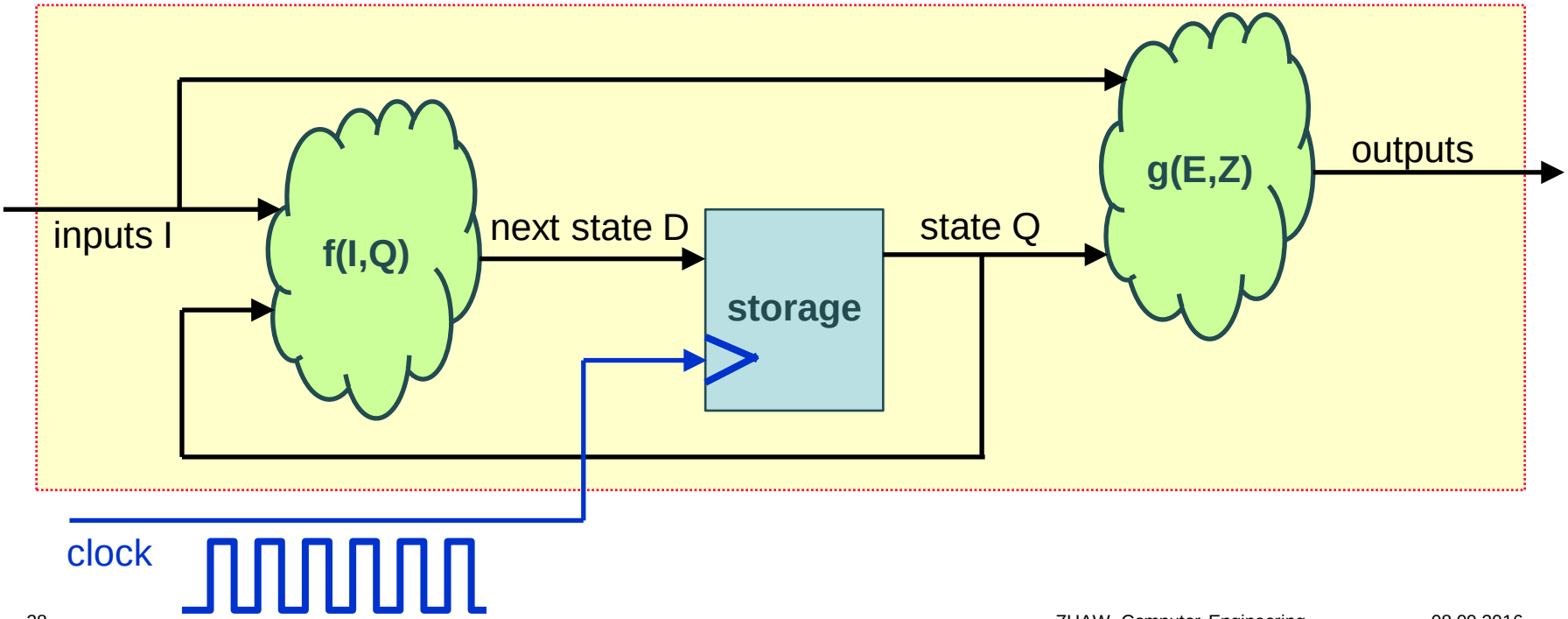
current state	next state
red	yellow A
yellow A	green
green	yellow B
yellow B	red





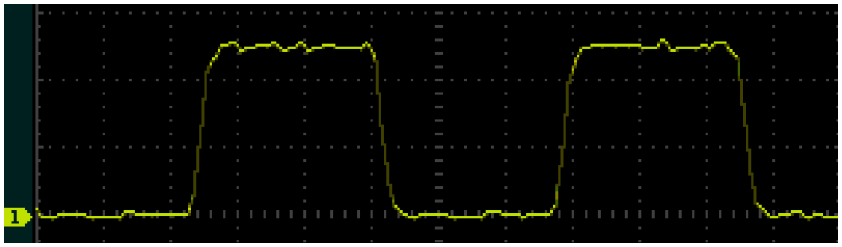
Sequential Logic

- General form → Finite State Machine (FSM)
  - Contains memory, i.e. storage of system state
  - Outputs depend on inputs and internal state
  - Next system state depends on current state and inputs → clock
- We will start with the analysis of simpler forms

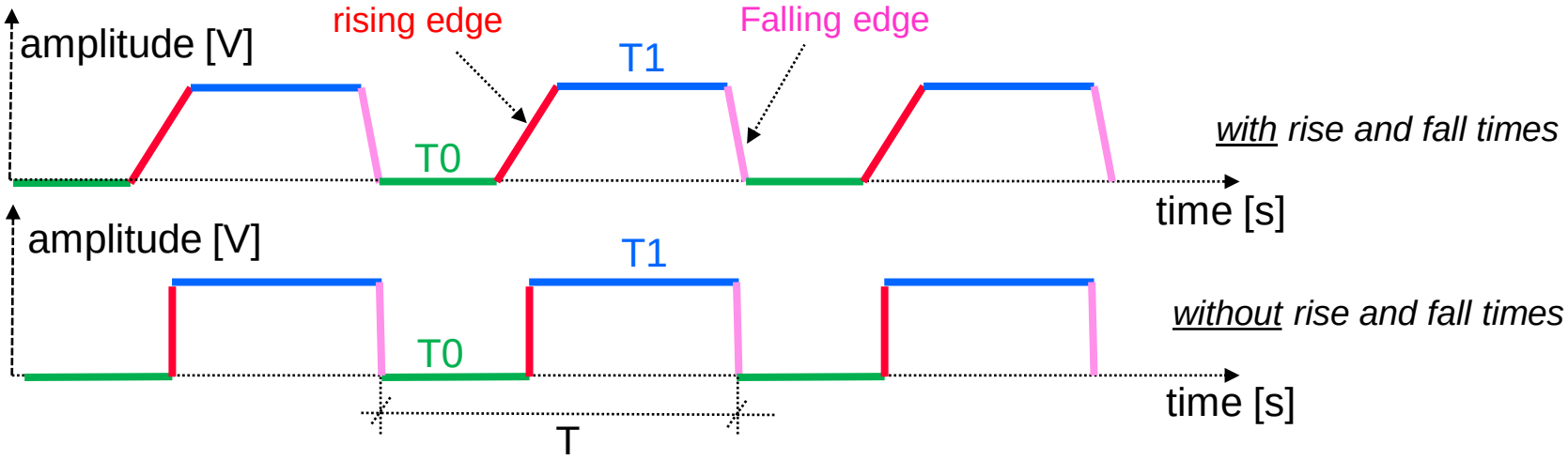


Clock Signal

- **Measured timing**



- **Abstract timing representation**



period T = T0 + T1 [s] **frequency** f = 1/T [Hz] **duty cycle** = T1/T [-]

Clock Signal

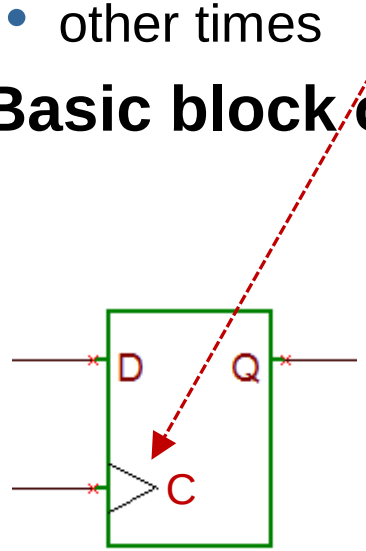
- **Period T**
  - measured in seconds (s)
- **Frequency f**
  - measured in Hertz (1/s)
  - i.e. number of cycles per second

T	f
1 s	1 Hz
1 ms = 10^{-3} s	1 kHz = 10^{3} Hz
1 us = 10^{-6} s	1 MHz = 10^{6} Hz
1 ns = 10^{-9} s	1 GHz = 10^{9} Hz

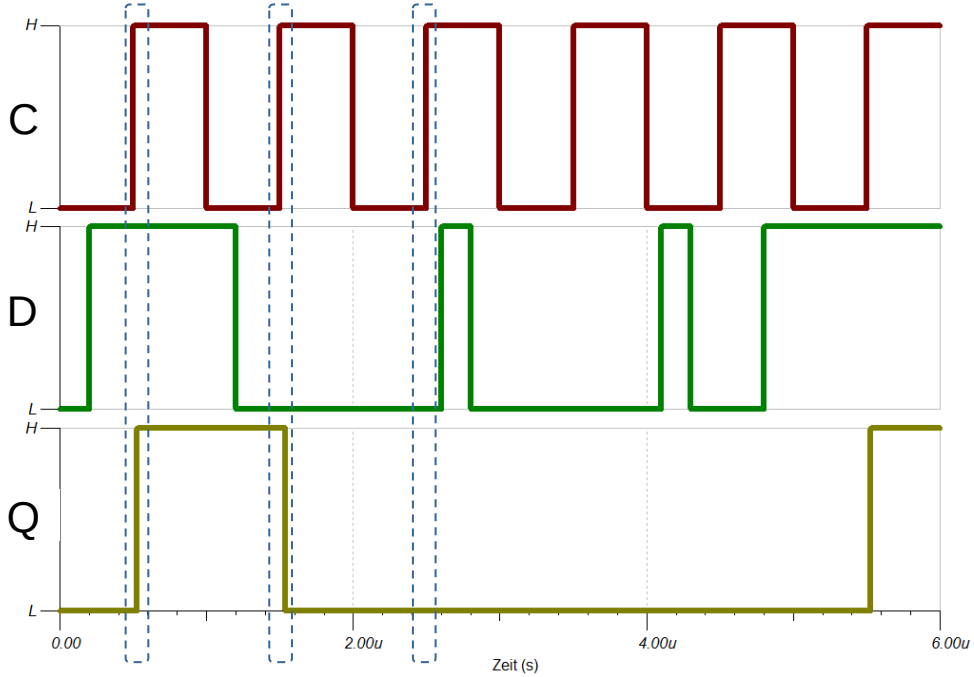
animation

D-Flip-Flop

- **Edge triggered storage element**
  - rising edge of C → current value at input D is stored (Q = D)
  - other times → no change of Q
- **Basic block of all our sequential circuits**



value at input D is stored and transferred to output Q, if C changes from 0 to 1 (0 → 1)



animation

D-Flip-Flop

School of Engineering
InES Institute of Embedded Systems

- **One flip-flop can represent two states**

Q	state
0	S0
1	S1

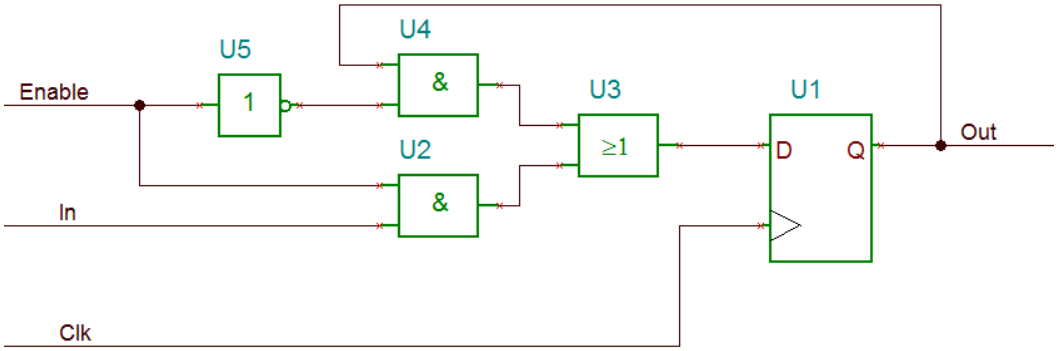
- **Two flip-flops can represent four states**

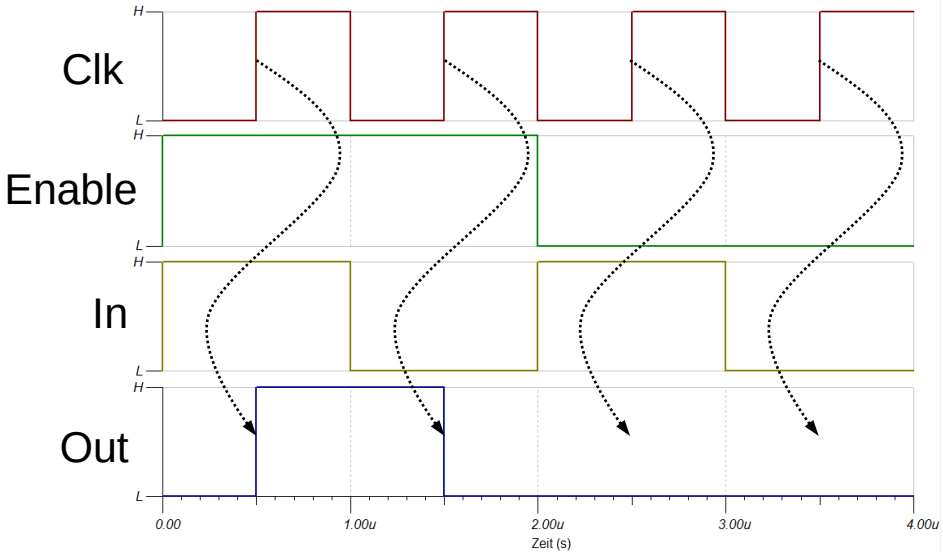
Q1	Q0	state
0	0	S0
0	1	S1
1	0	S2
1	1	S3

- **n flip-flops can represent 2^n states**

Timing Diagram

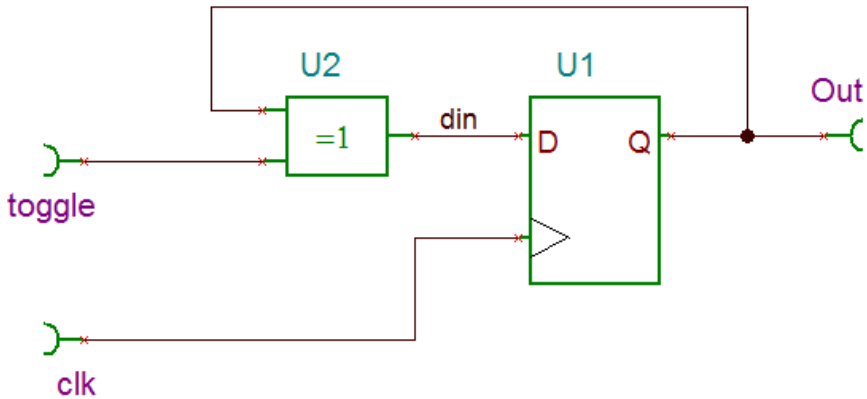
- **Example: Flip-flop with multiplexer**

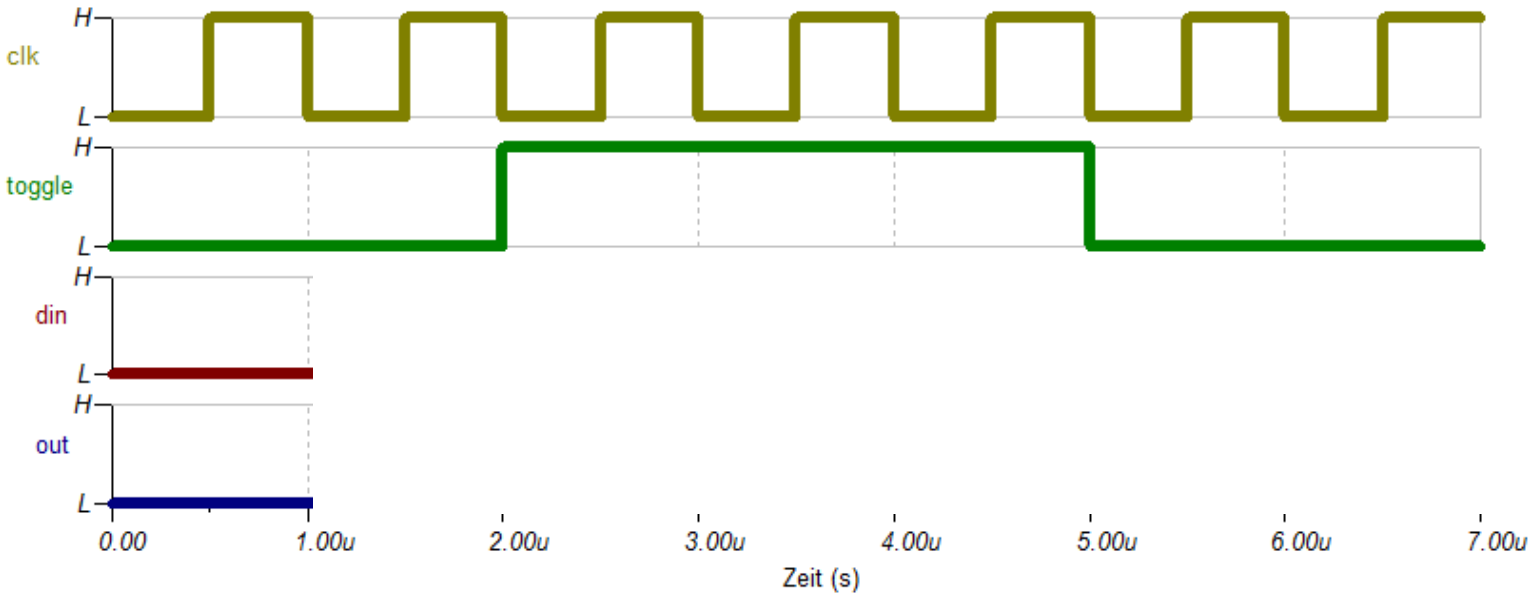




Timing Diagram

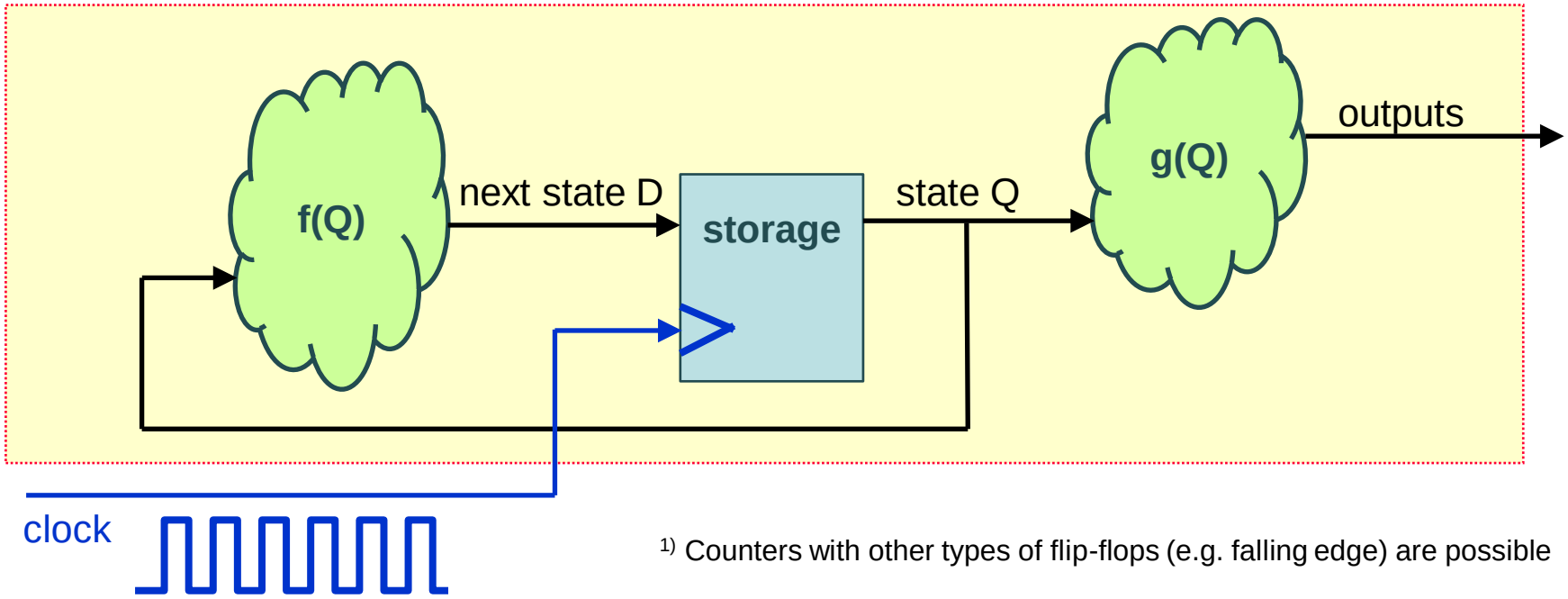
- **Exercise**
  - Complete the timing diagram





Counter

- Simple form of sequential logic (finite state machine)
- State changes with rising clock edge [1)]
- Next state depends only on current state
  - Sequence of states cannot be influenced from the outside
- Outputs depend only on internal state, not on any inputs

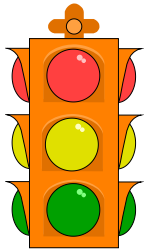


[1)] Counters with other types of flip-flops (e.g. falling edge) are possible

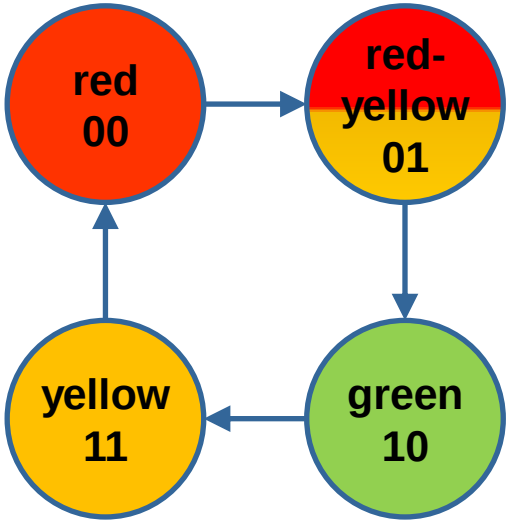
Counter

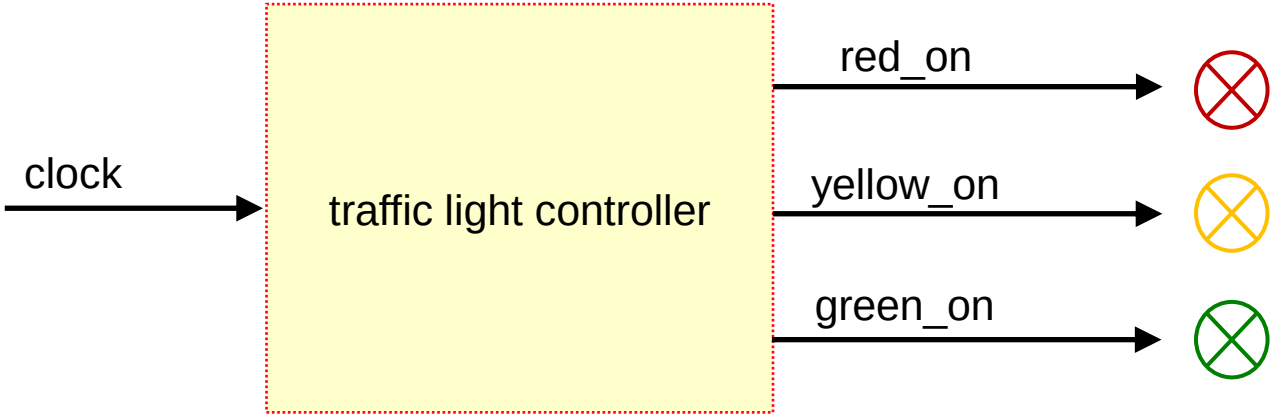
School of Engineering
InES Institute of Embedded Systems

- **Traffic light: Encoding of states**
  - Encoding is selected by circuit designer



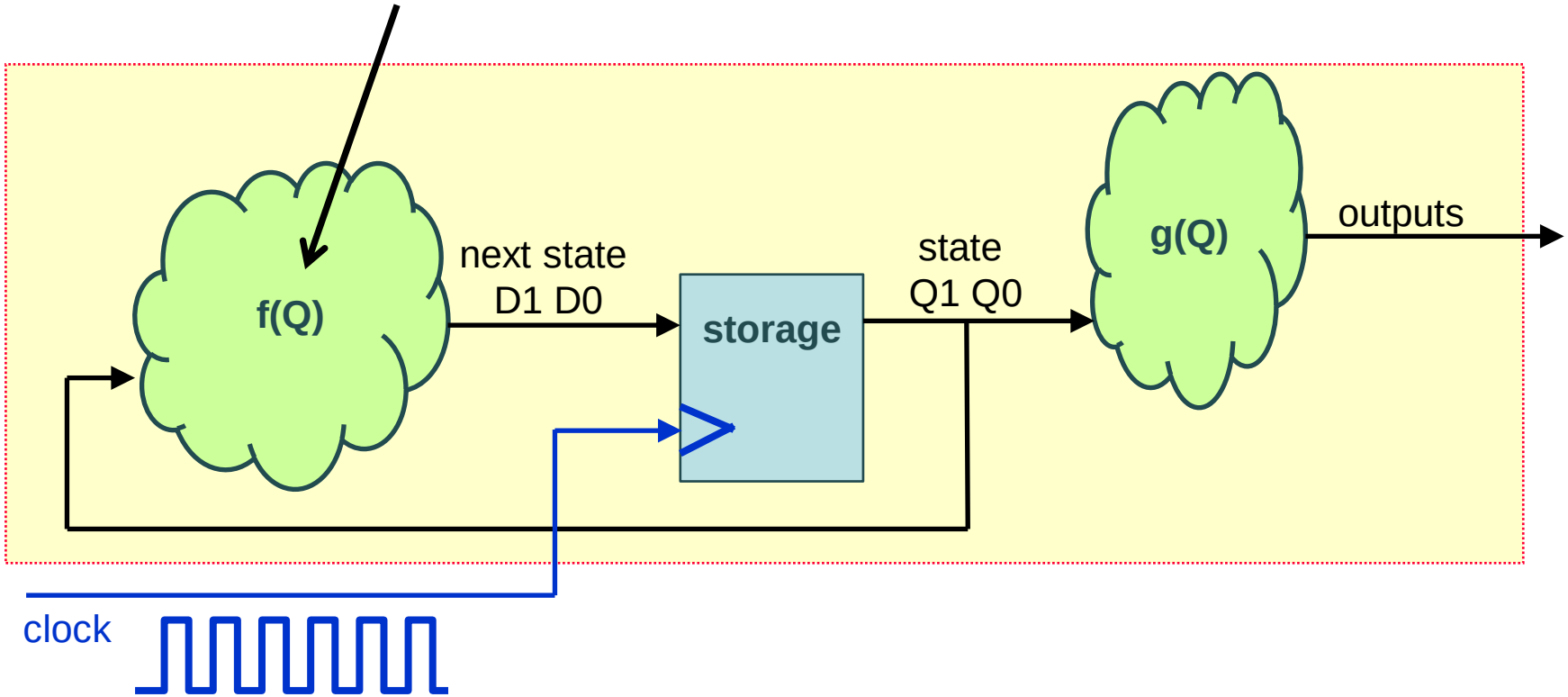
current state Q		next state D	
00	red	red-yellow	01
01	red-yellow	green	10
10	green	yellow	11
11	yellow	red	00



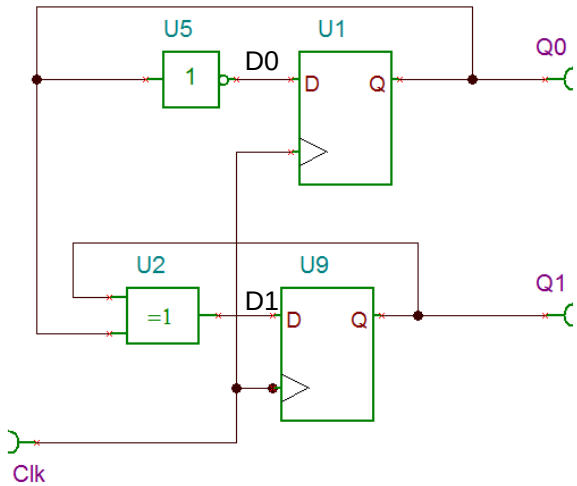


Counter

- **Traffic Light: Next state ???**
  - function of current state



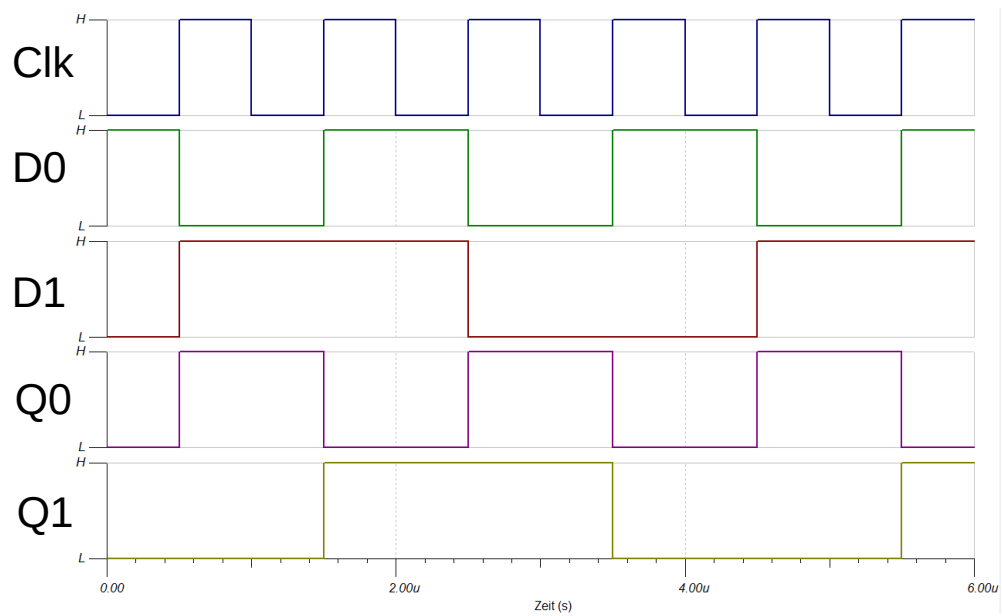
Counter: 2-bit Binary Counter for Traffic Light



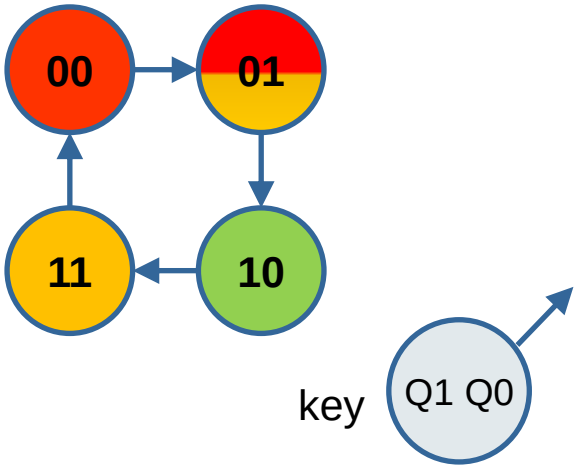
D1 = Q0 $ Q1
D0 = !Q0

Q1	Q0	D1	D0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

timing diagram

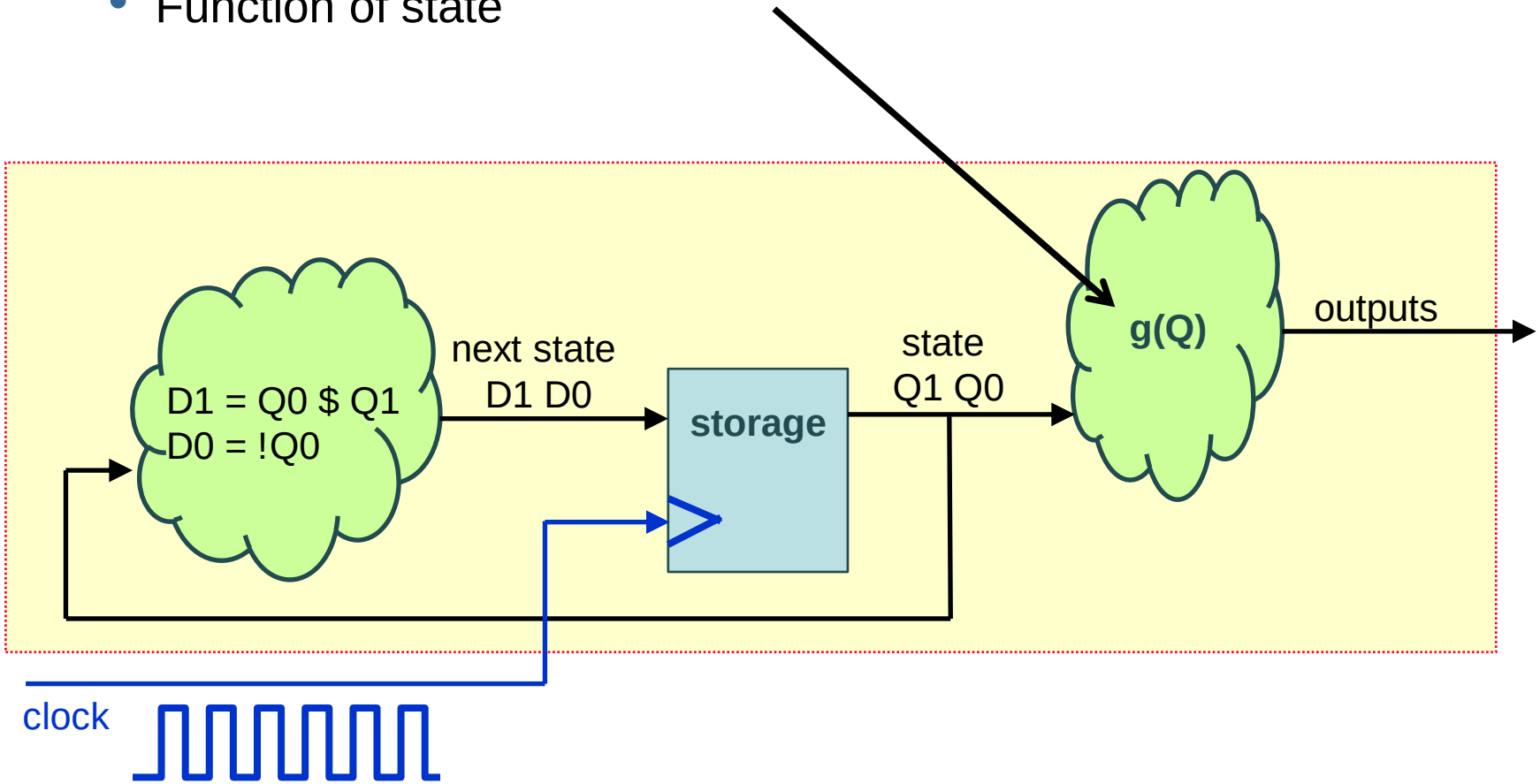


state diagram



Counter

- **Traffic Lights: Outputs ???**
  - Function of state

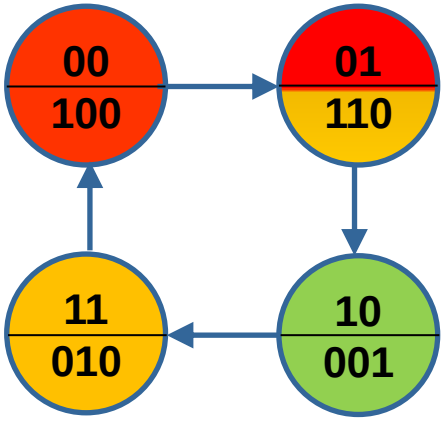


Counter

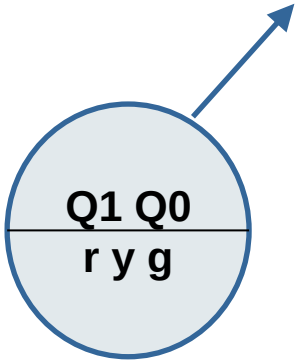
- **Traffic light**

state		outputs		
Q1	Q0	red_on	yellow_on	green_on
0	0	1	0	0
0	1	1	1	0
1	0	0	0	1
1	1	0	1	0

state diagram

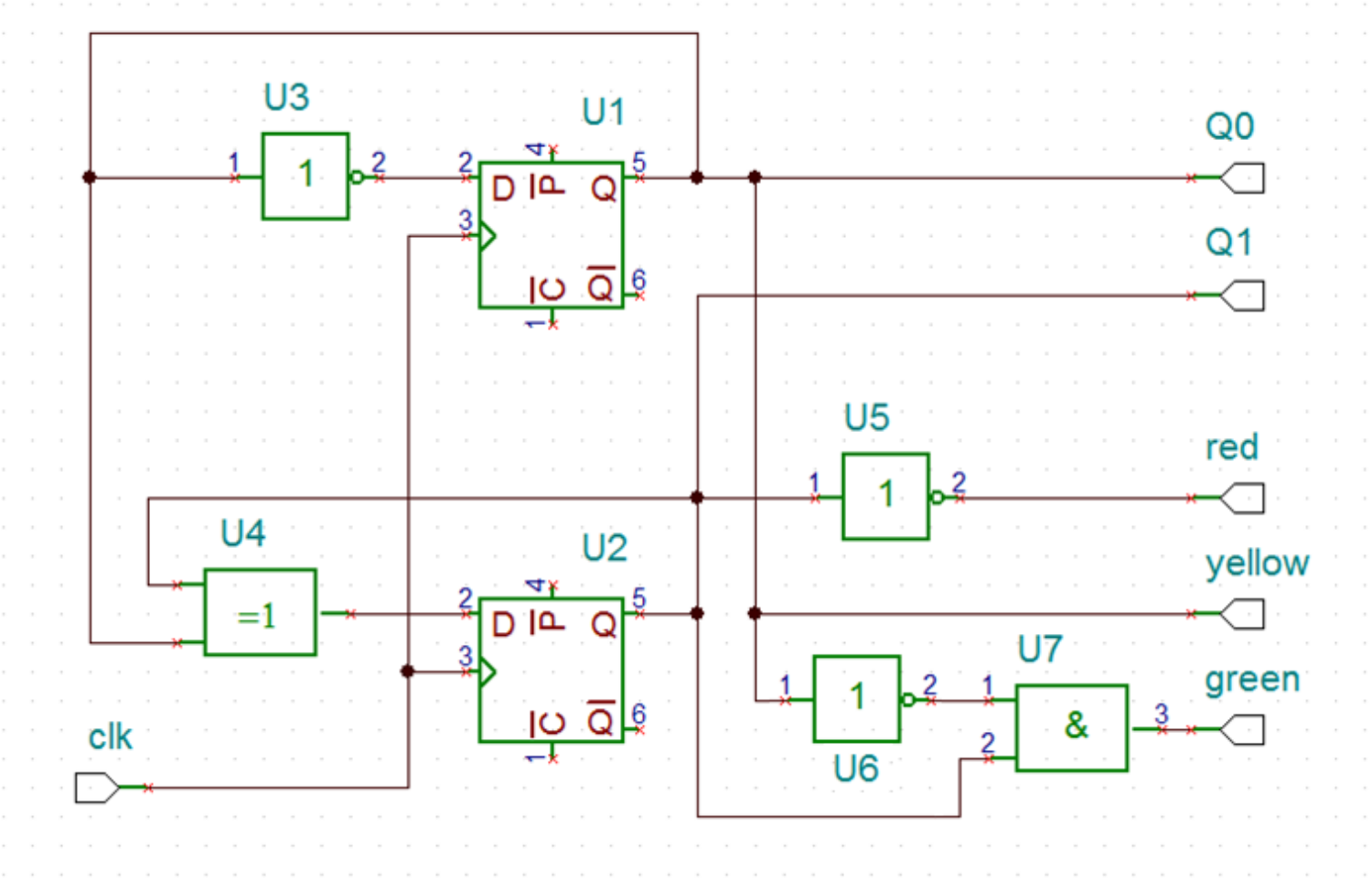


key



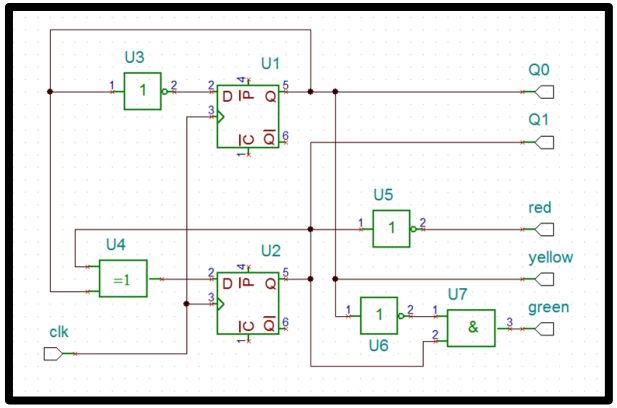
Counter

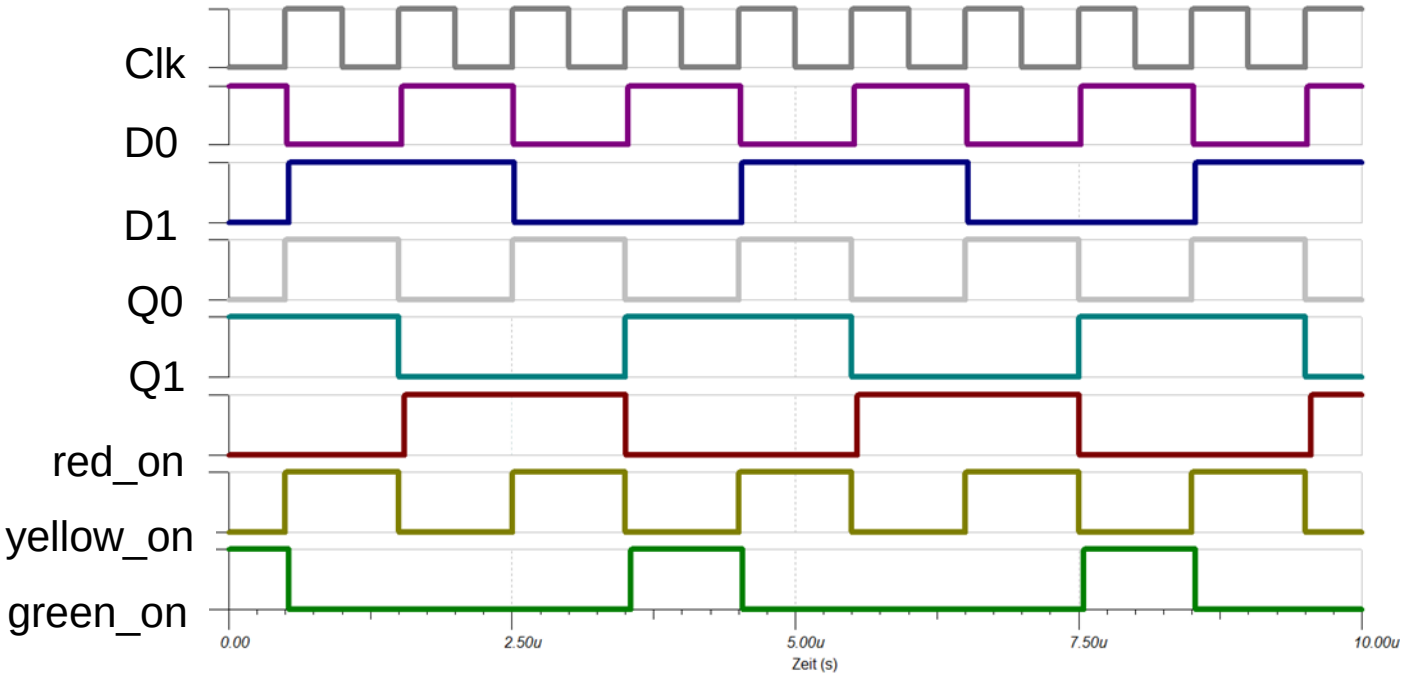
- **Traffic light: schematic**



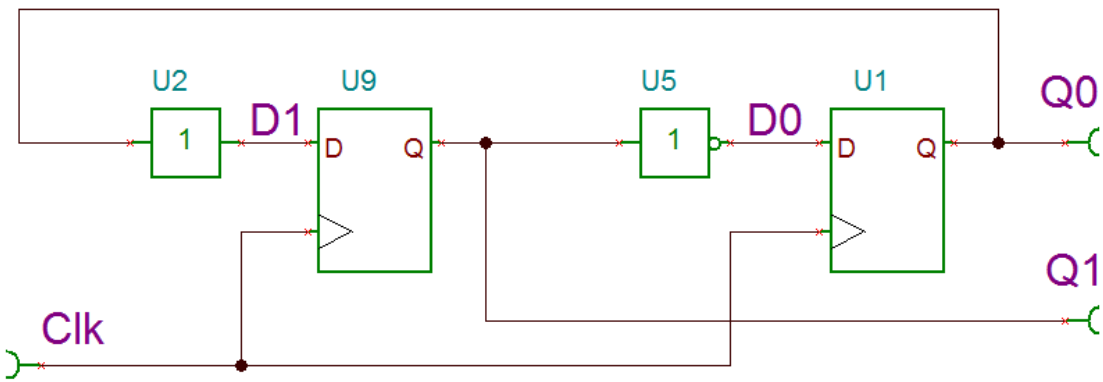
Counter

- **Traffic light: timing diagram**



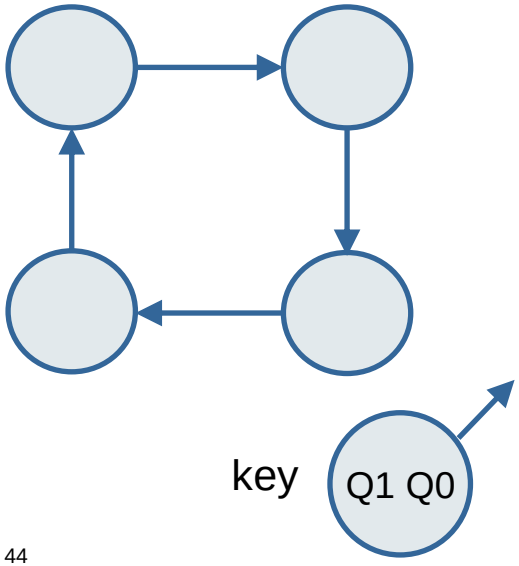


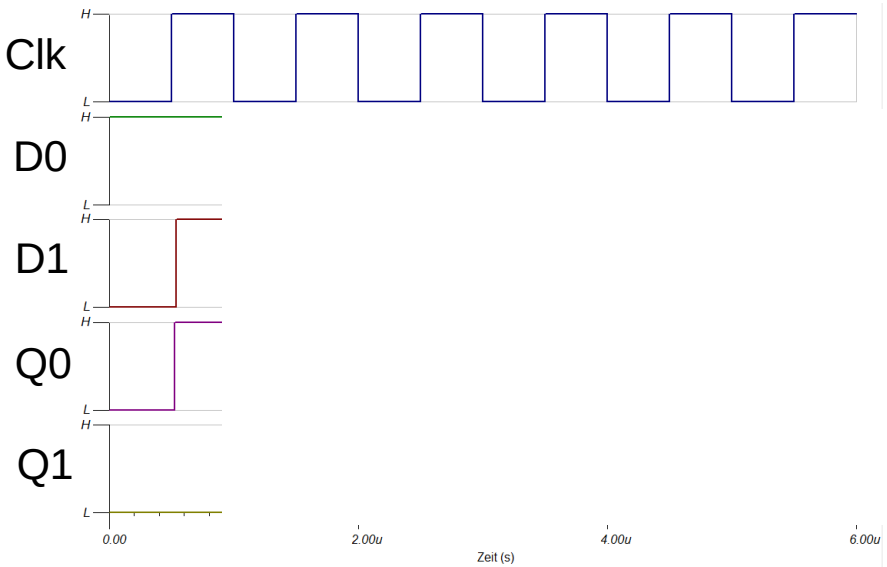
Counter: 2-bit Gray Counter



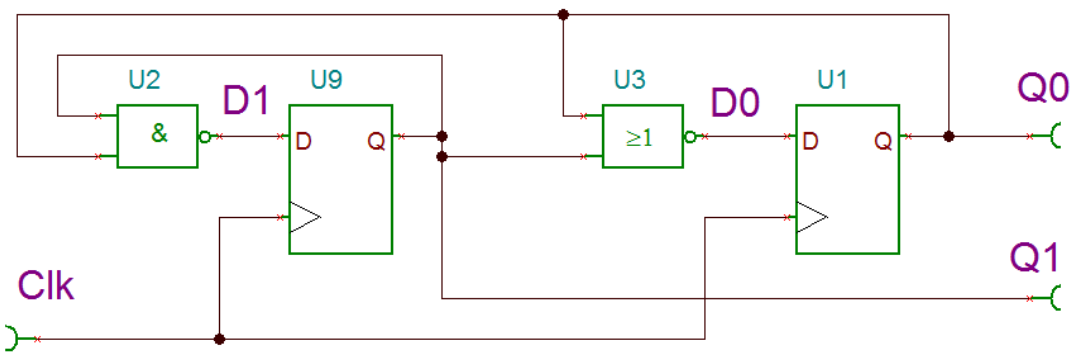
Q1	Q0	D1	D0
0	0		
0	1		
1	0		
1	1		

Exercise: Complete the truth table, the state diagram and the timing diagram



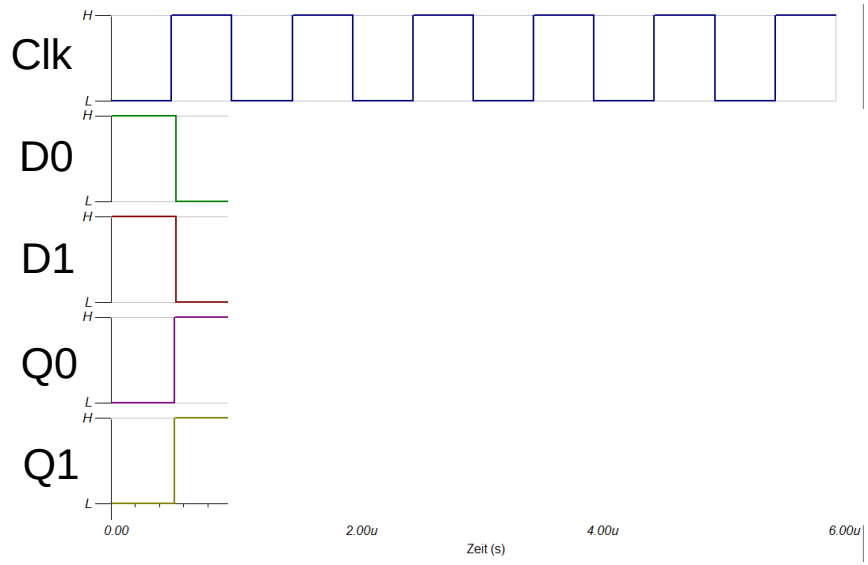


Counter: Another example

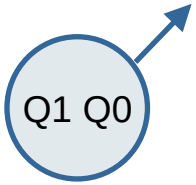


Q1	Q0	D1	D0
0	0		
0	1		
1	0		
1	1		

Exercise: Fill out the truth table, the state diagram and the timing diagram

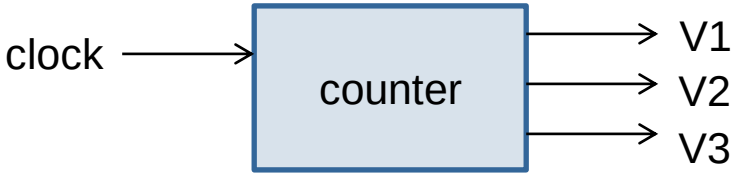


key



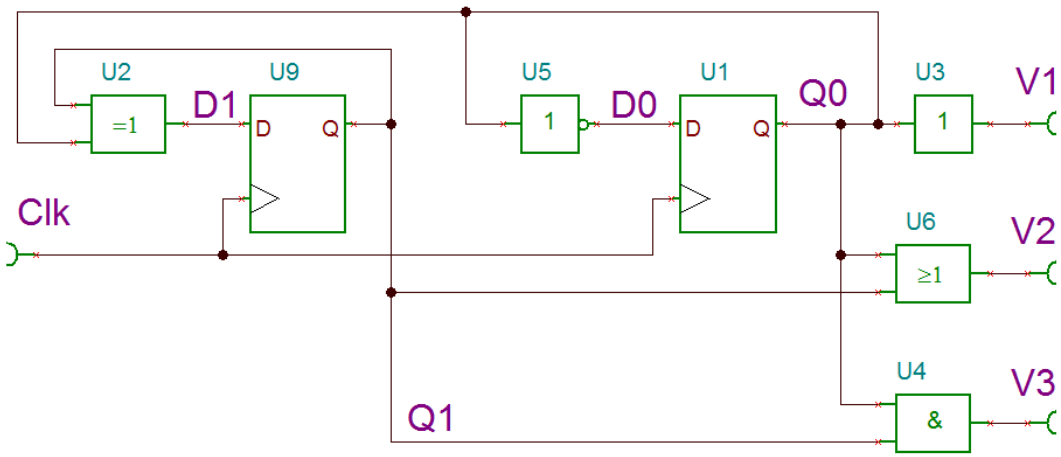
Counter: Water Fountain

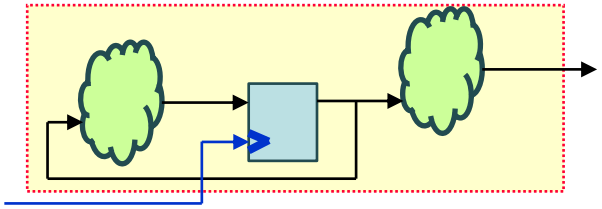
- Water fountain with three valves
  - Counter controls valves V1 to V3



- Valve open, i.e. water flows
  - Output Vx = '1'

Counter: Water fountain





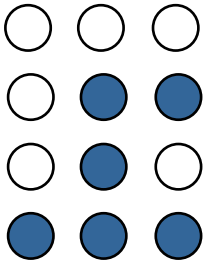
state diagram

→ same as binary counter

Q1	Q0	D1	D0	V3	V2	V1
0	0	0	1	0	0	0
0	1	1	0	0	1	1
1	0	1	1	0	1	0
1	1	0	0	1	1	1

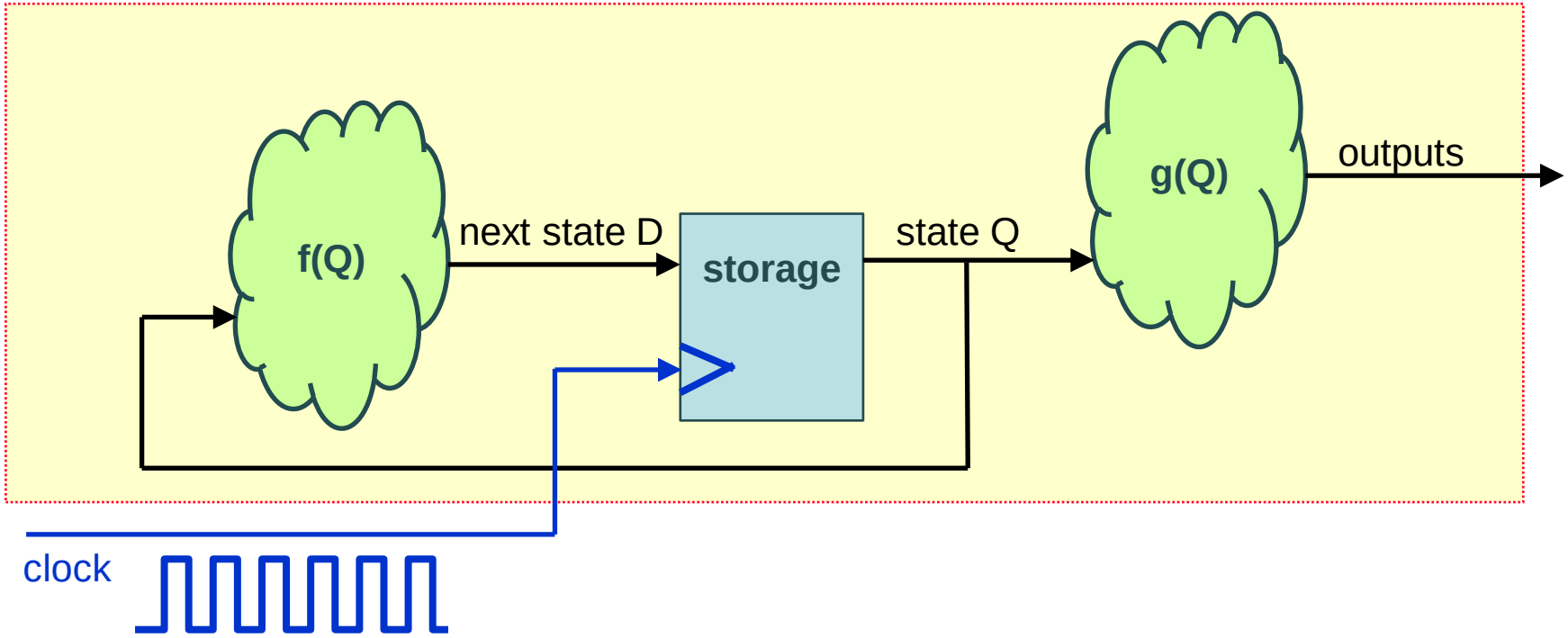
pattern at the fountain

V3 V2 V1



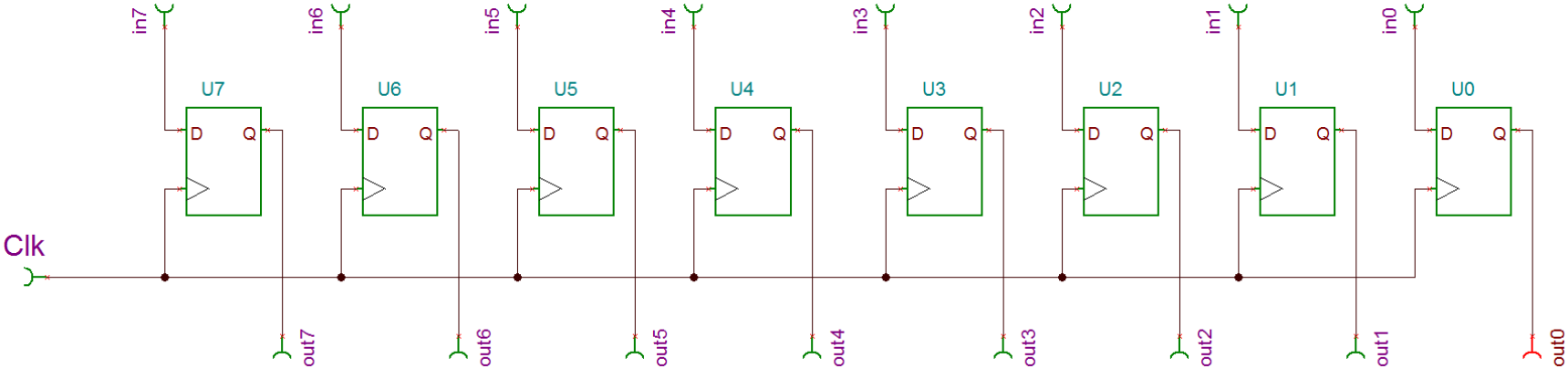
Counter

- Summary counter
  - One or several flip-flops to store the state
  - All flip-flops on the same clock
  - Sequence of states cannot be influenced from the outside
  - Outputs depend only on the internal state



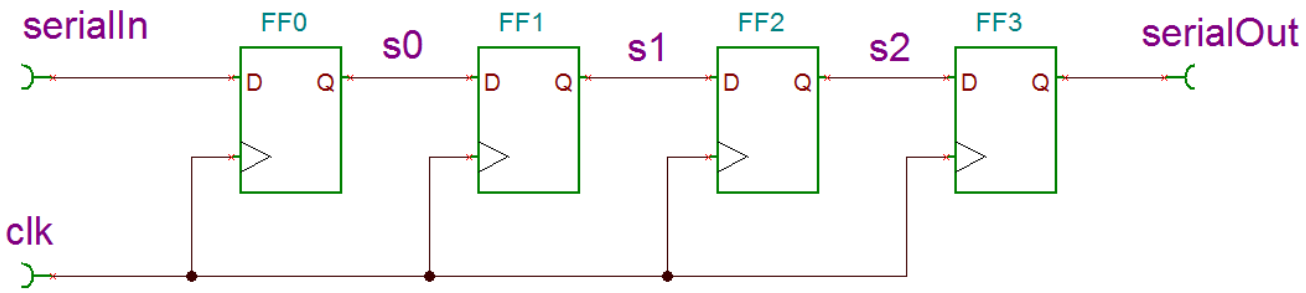
Register

- **(Parallel) register**
  - input and output are parallel
- **Example 8-bit register**
  - inputs in7 – in0 → registered on clock edge
  - outputs out7 – out0



Shift Register

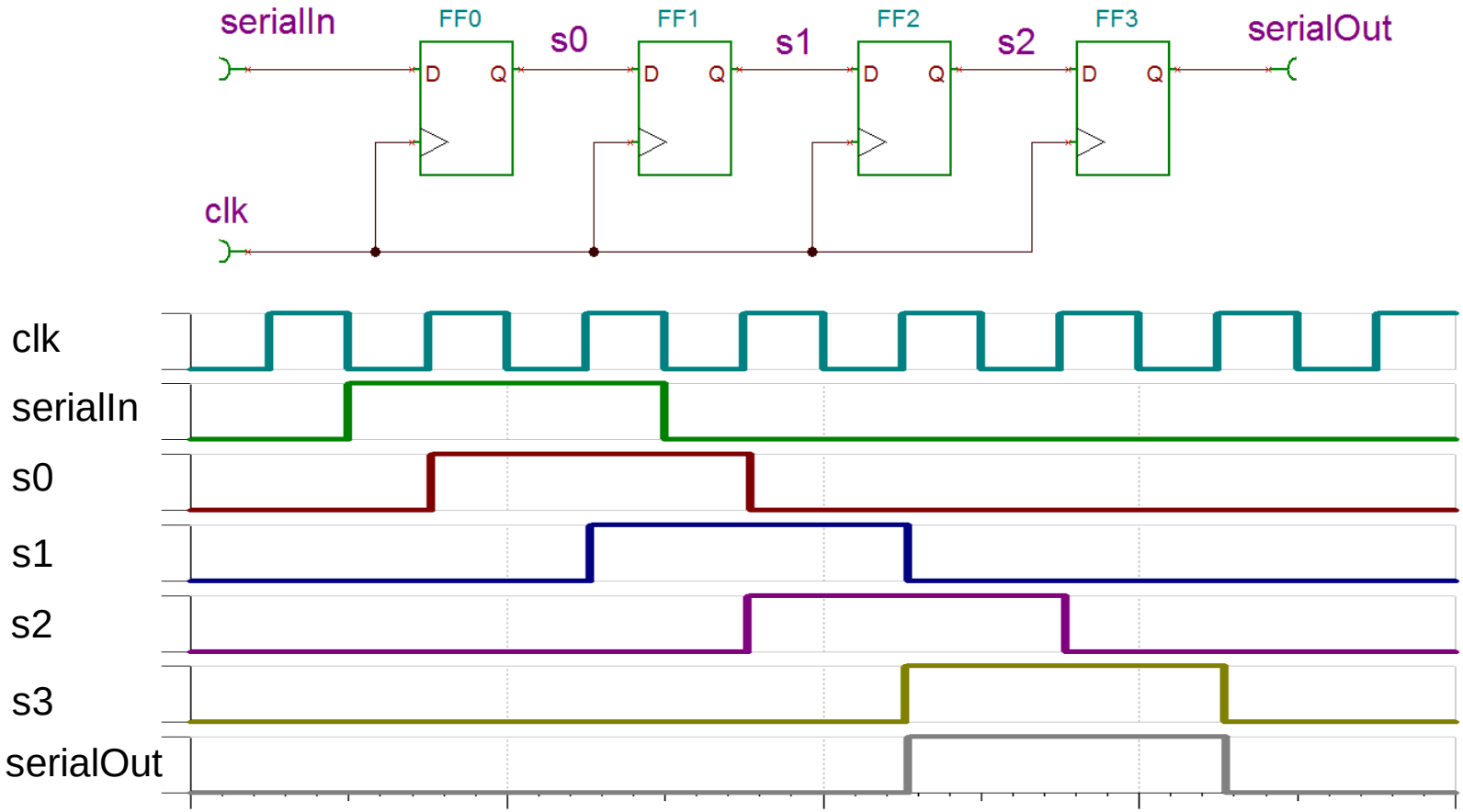
- **Chain of connected D-flip-flops**
  - Output of FFX is connected to input of FFX+1
  - Input of first FF → serial input of shift register
  - Output of last FF → serial output of shift register
  - Often parallel reading of data possible through s0, s1, …, sx
  - Parallel write requires multiplexer on each FF input



Example of a 4-bit shift register

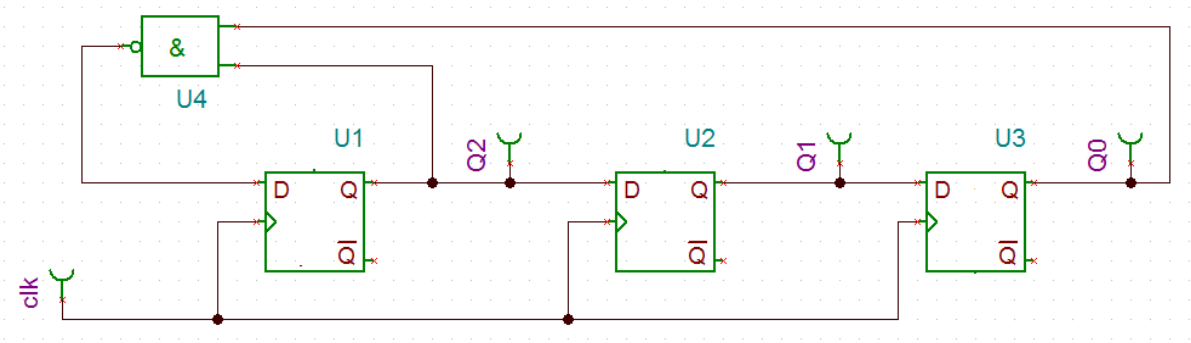
Shift Register

- **Timing diagram**
  - Input pattern is repeated with a delay on the output of each FF



Shift Register with Feedback

- Output is fed back to the input through a logic function



Q2	Q1	Q0	D2	D1	D0
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Exercise
- Fill in the truth table
- Draw the state diagram

Shift Registers: Applications

School of Engineering
InES Institute of Embedded Systems

- **Ethernet and USB**
  - convert serial bit streams to parallel and vice versa
  - serial requires less connections
  - but processer works on parallel data
- **Mobile phones, Ethernet, miscellaneous interfaces**
  - shift register with feedback for error detection
  - Processors (JTAG, scan chains, …)
  - program development
    - set breakpoints
    - monitor internal states
  - verification
  - production test → does each transistor / gate work?

Conclusion

- **Sequential logic → Finite State Machine (FSM)**
  - system has memory → storage of system state
- **Counter**
  - state changes with clock
  - new state is predetermined. No control from the outside
- **Moore machine**
  - state influenced by inputs
- **Description of system behavior**
  - truth table
  - state diagram
  - timing diagram
- **Register and shift register**