Finite State Machine (1A)

Young Won Lim
6/6/18

Please send corrections (or suggestions) to youngwlim@hotmail.com.

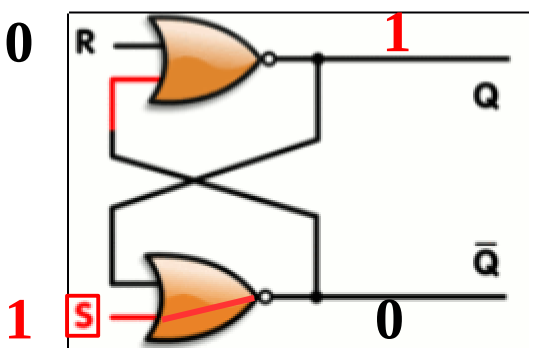
This document was produced by using LibreOffice and Octave.

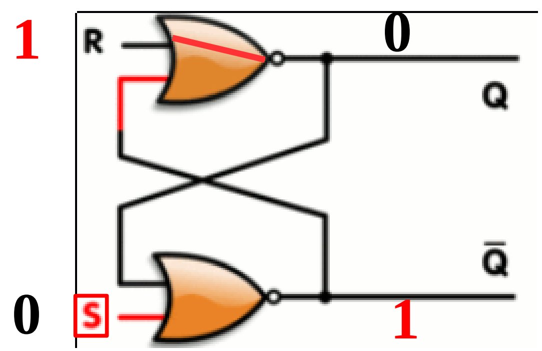
FSM and Digital Logic Circuits

- Latch
- D FlipFlop
- Registers
- Timing
- Mealy machine
- Moore machine
- Traffic Lights Examples

https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

NOR-based SR Latch – SET / RESET





SET S=**1** R=0

Q=1

$\overline{Q}$=0

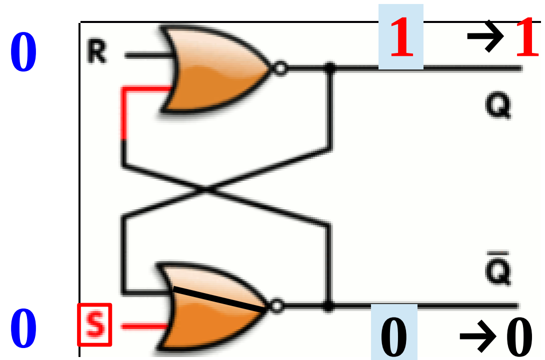
RESET S=0 R=**1**

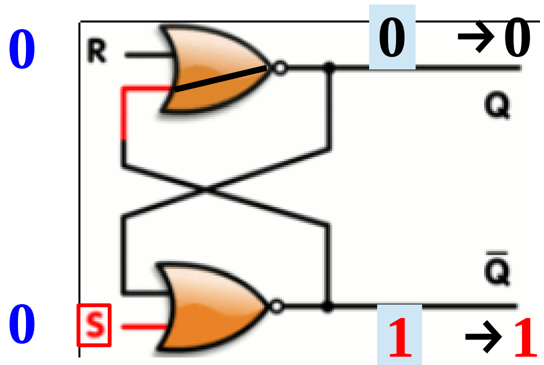
Q=0

$\overline{Q}$=1

https://en.wikipedia.org/wiki/Flip-flop_(electronics)

NOR-based SR Latch – HOLD





HOLD	S=0 R=0

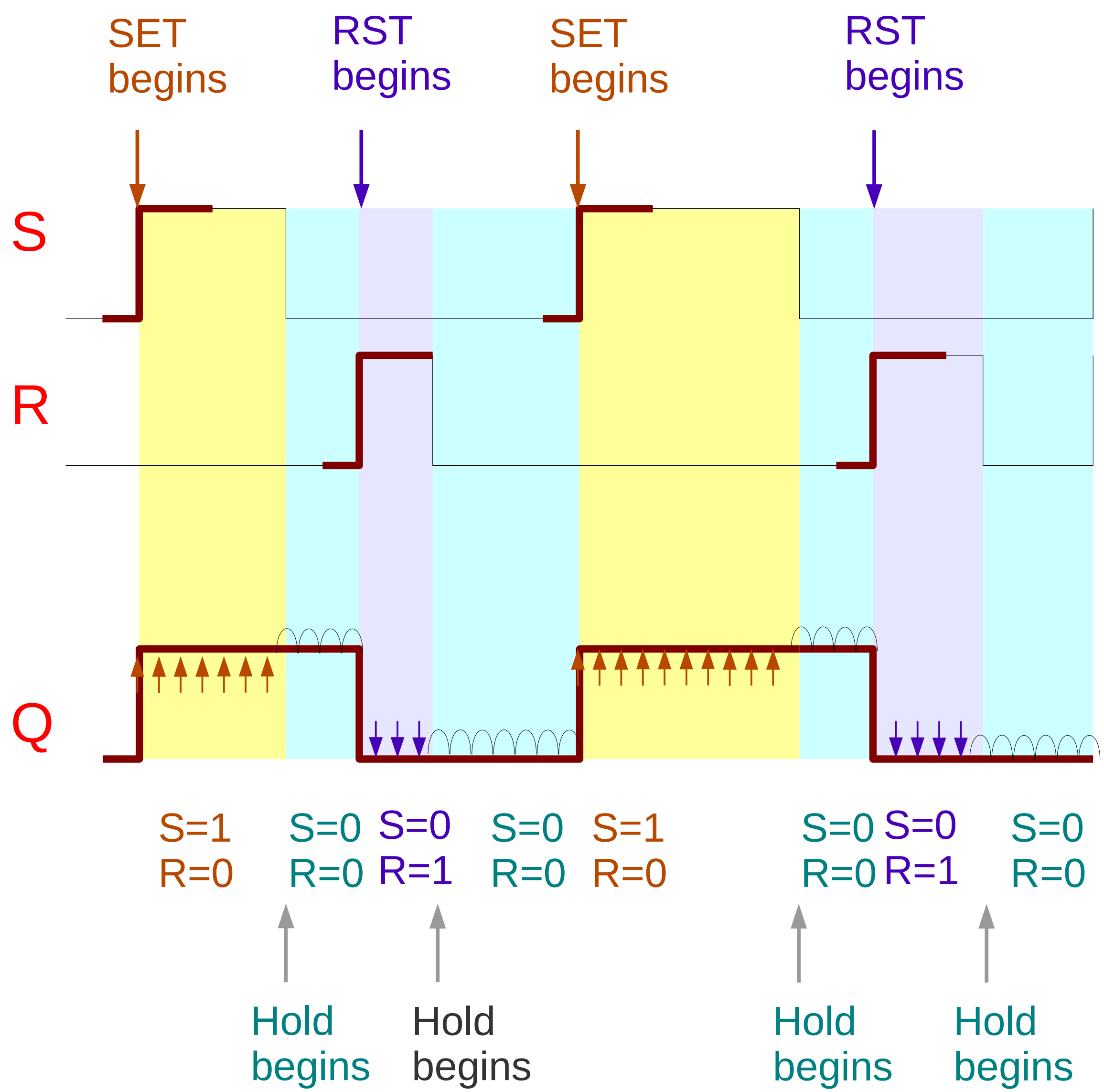
Q=old Q
$\overline{Q}$=old $\overline{Q}$

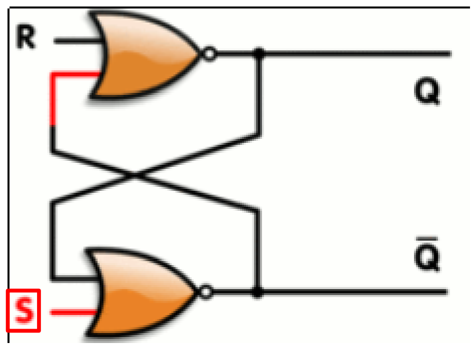
HOLD	S=0 R=0

Q=old Q
$\overline{Q}$=old $\overline{Q}$

https://en.wikipedia.org/wiki/Flip-flop_(electronics)

NOR-based SR Latch

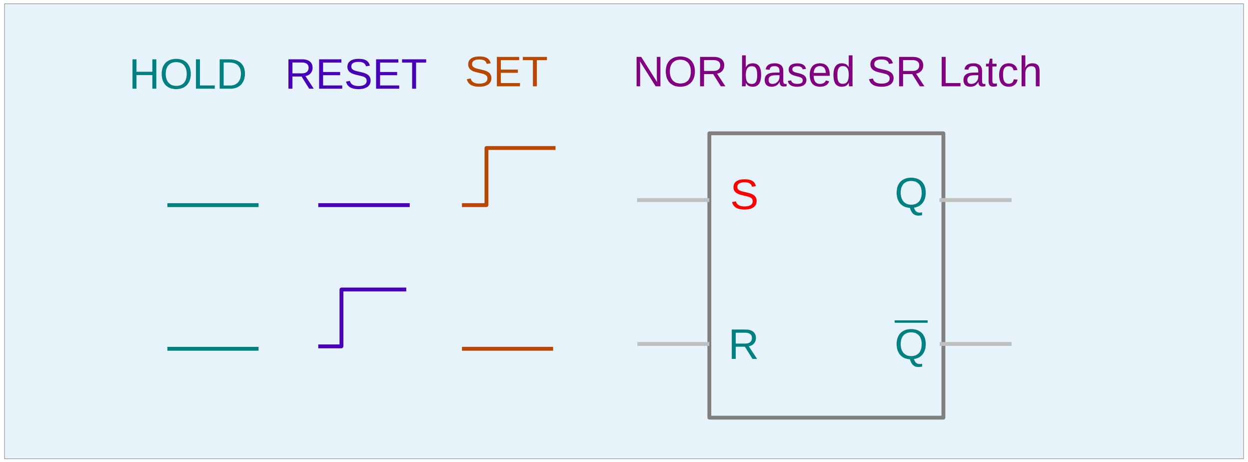


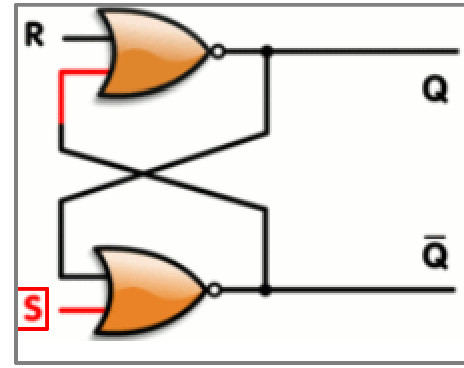


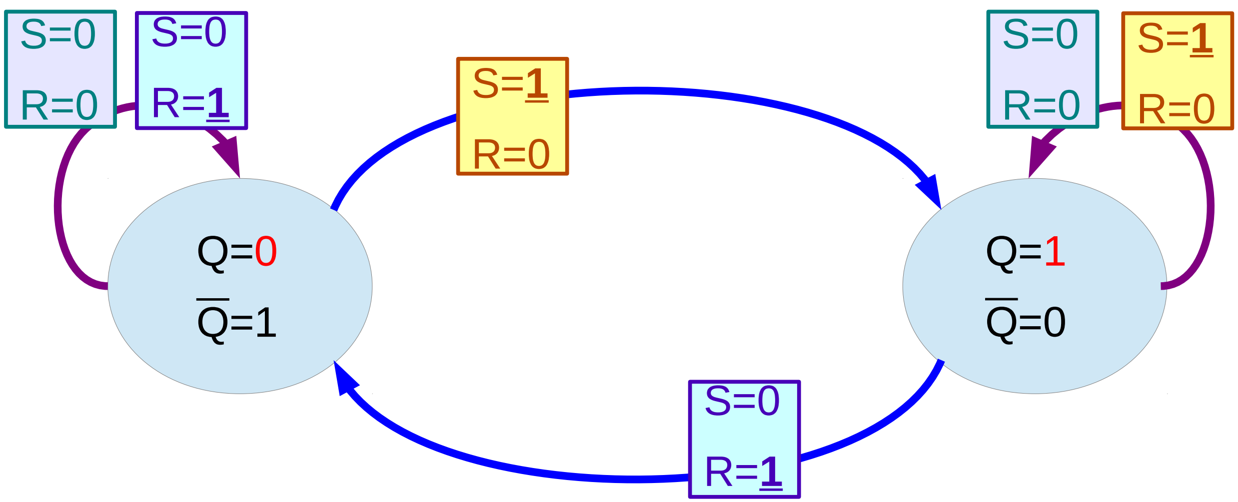
SET	S=**1** R=0	Q=1 $\overline{Q}$=0
RESET	S=0 R=**1**	Q=0 $\overline{Q}$=1
HOLD	S=0 R=0	Q=old Q $\overline{Q}$=old $\overline{Q}$

https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

NOR-based SR Latch States



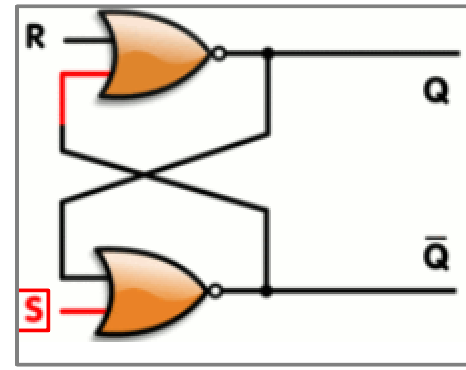


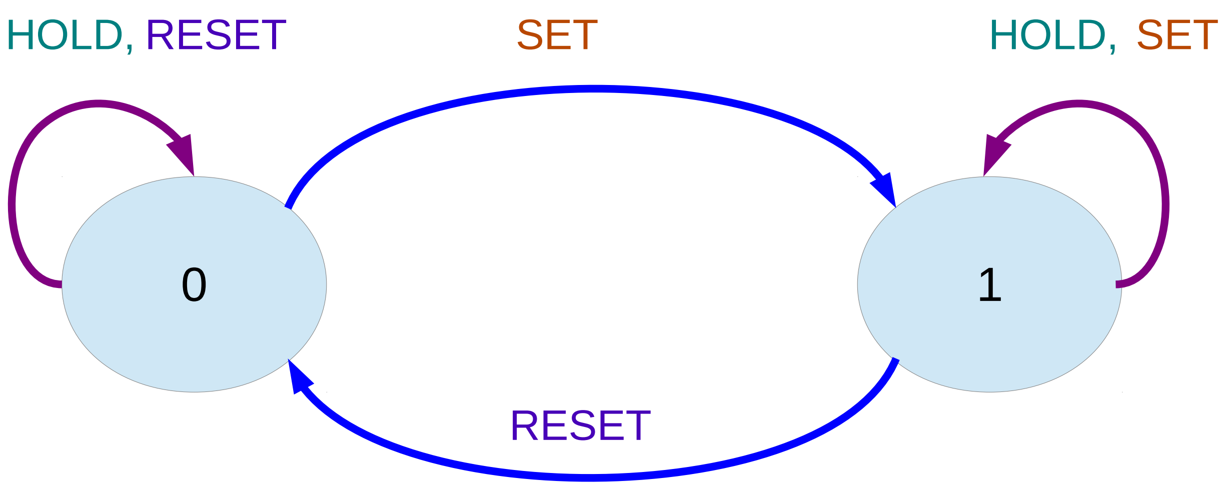


SET	S=1 R=0	Q=1 $\overline{Q}$=0
RESET	S=0 R=1	Q=0 $\overline{Q}$=1
HOLD	S=0 R=0	Q=old Q $\overline{Q}$=old $\overline{Q}$

https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

SR Latch States





SET	S=**1** R=0	Q=1 $\overline{Q}$=0
RESET	S=0 R=**1**	Q=0 $\overline{Q}$=1
HOLD	S=0 R=0	Q=old Q $\overline{Q}$=old $\overline{Q}$

https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

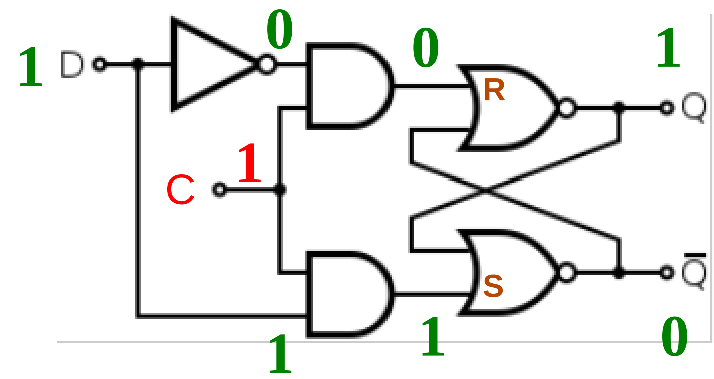
NOR-based D Latch – SET / RESET

https://en.wikipedia.org/wiki/Flip-flop_(electronics)

D=**1**
C=1

SET S=**1** R=0

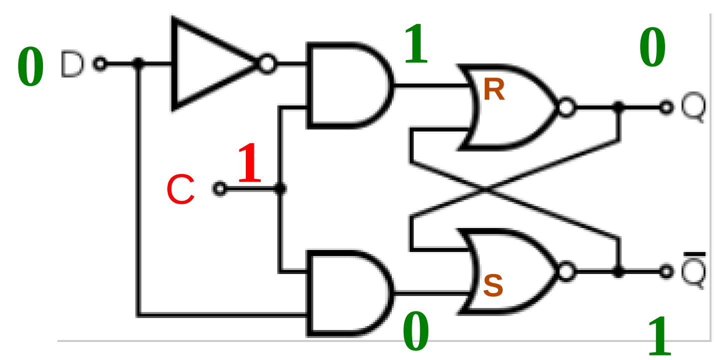
Q=1
$\overline{Q}$=0



D=**0**
C=1

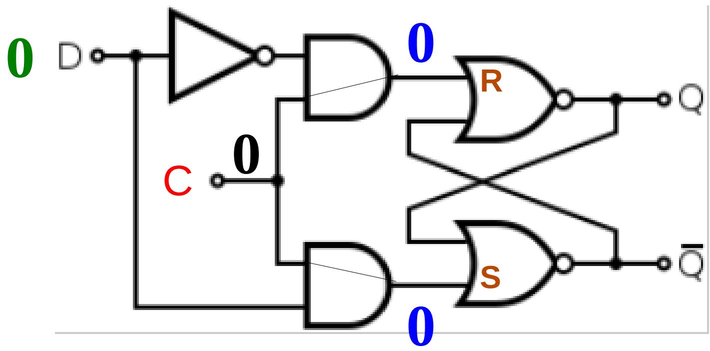
RESET S=0 R=**1**

Q=0
$\overline{Q}$=1



NOR-based D Latch – HOLD

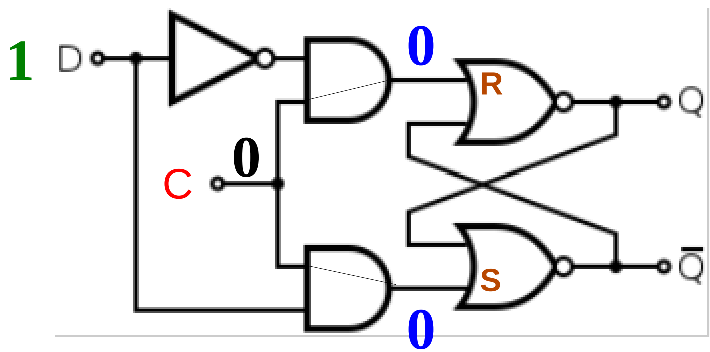
https://en.wikipedia.org/wiki/Flip-flop_(electronics)



D=**X**
C=0

HOLD
S=0
R=0

Q=old Q
$\overline{Q}$=old $\overline{Q}$

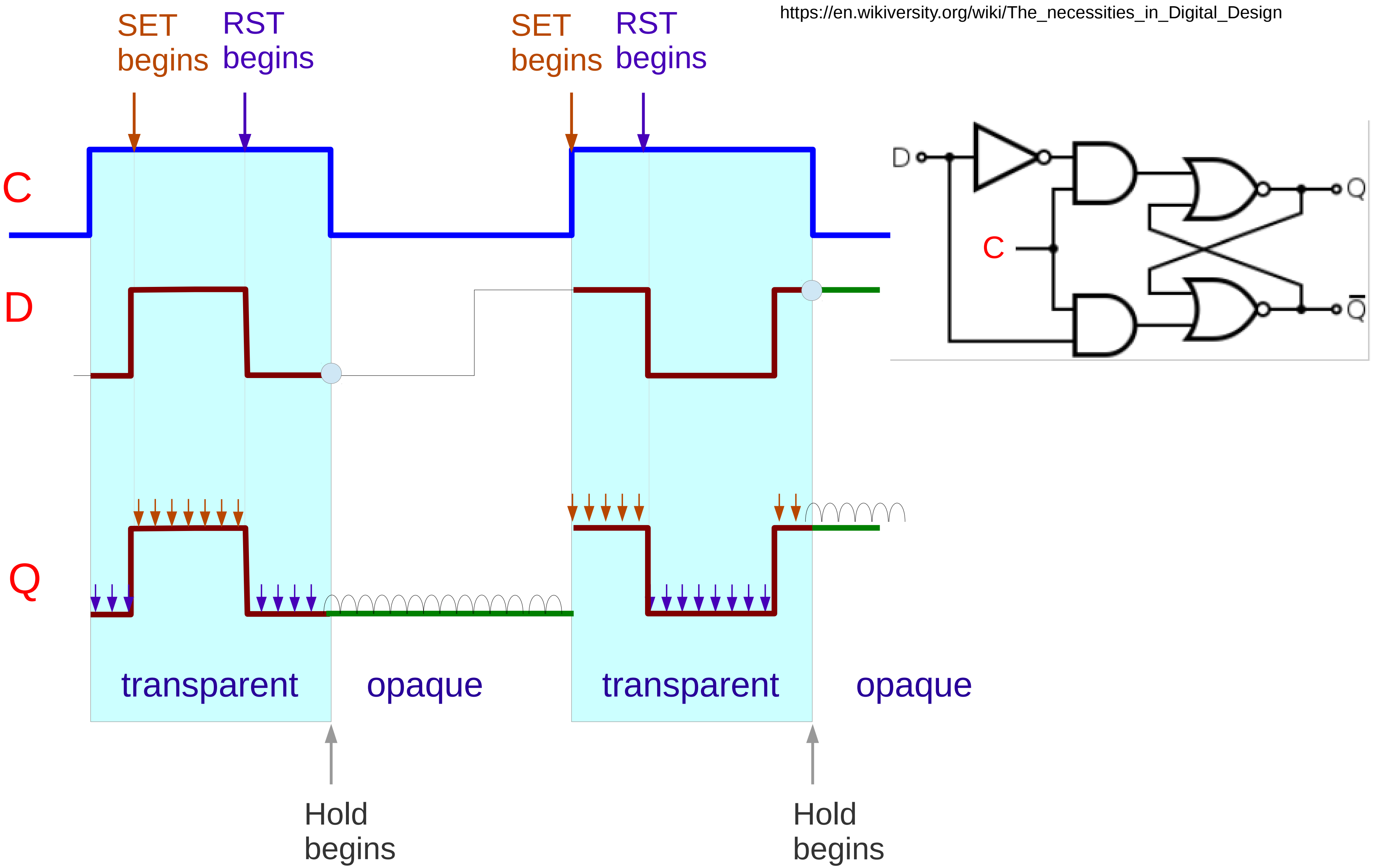


D=**X**
C=0

HOLD
S=0
R=0

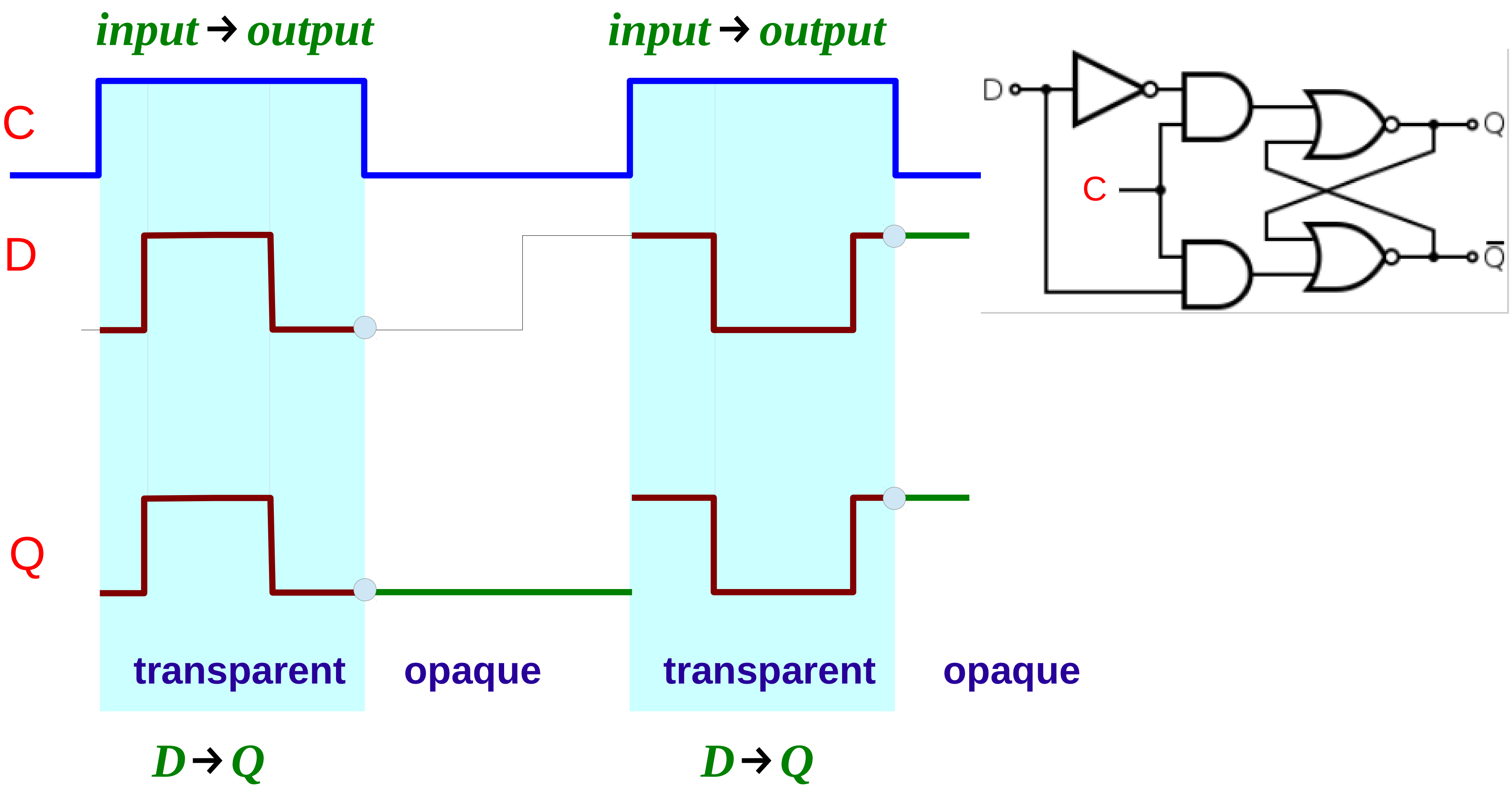
Q=old Q
$\overline{Q}$=old $\overline{Q}$

NOR-based D Latch – Set / Reset / Hold

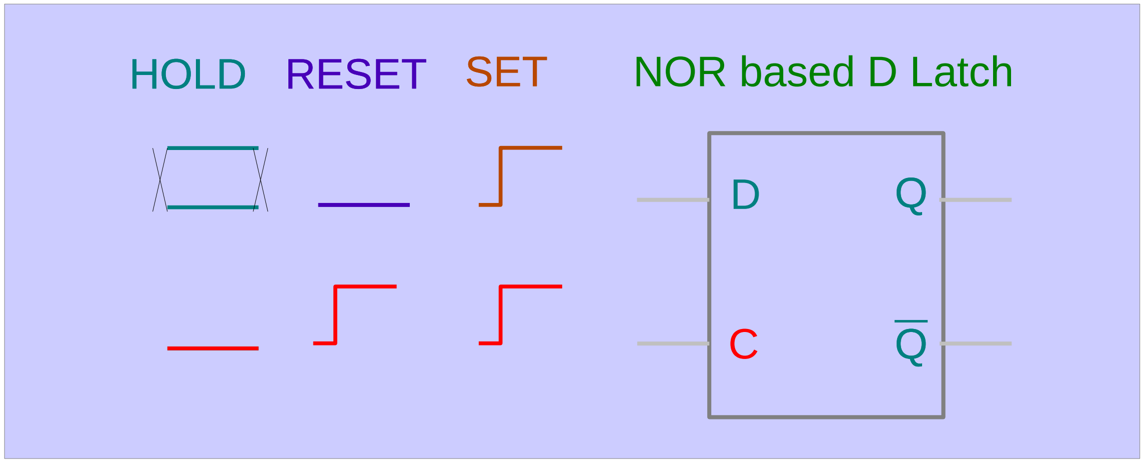


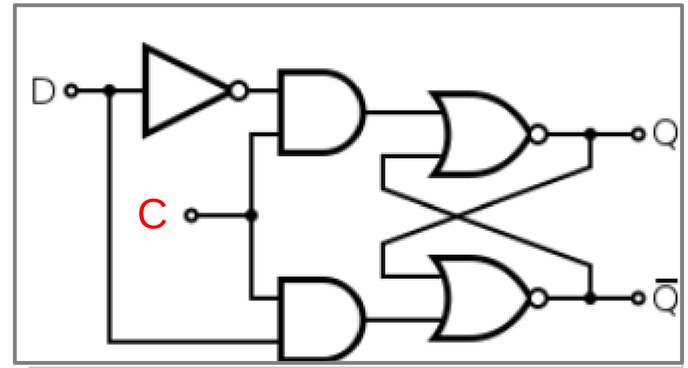
NOR-based D Latch – transparent / opaque

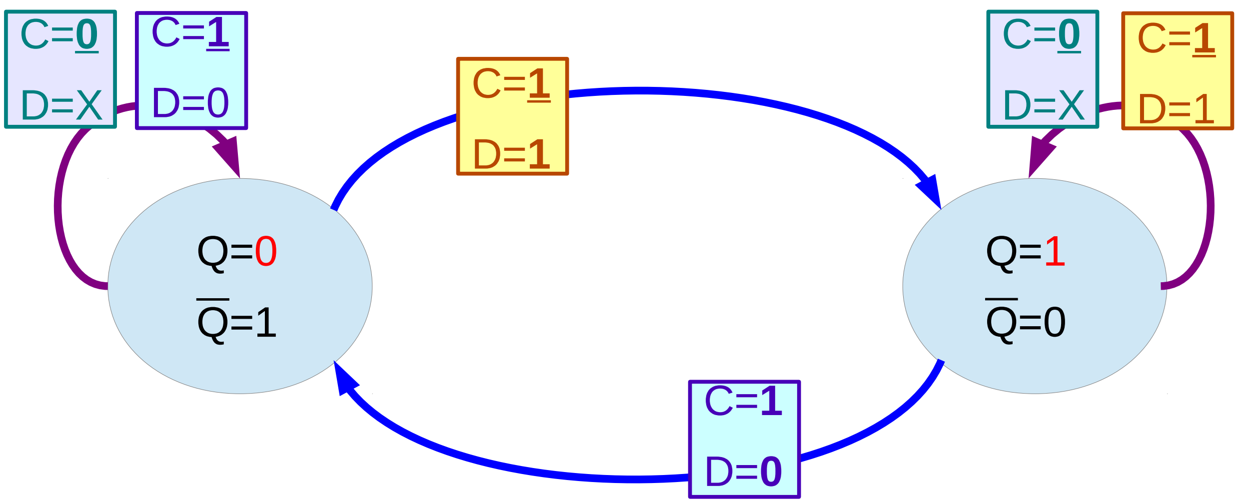
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design



NOR-based D Latch States

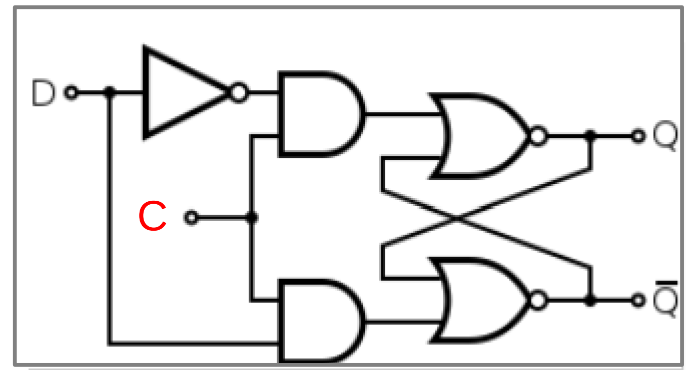


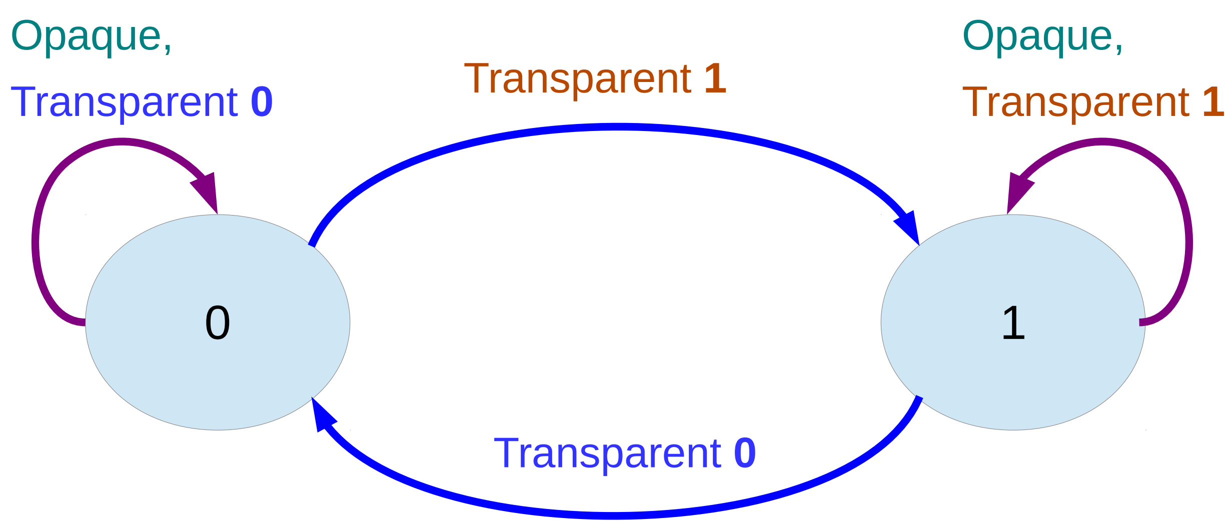




https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

D Latch States

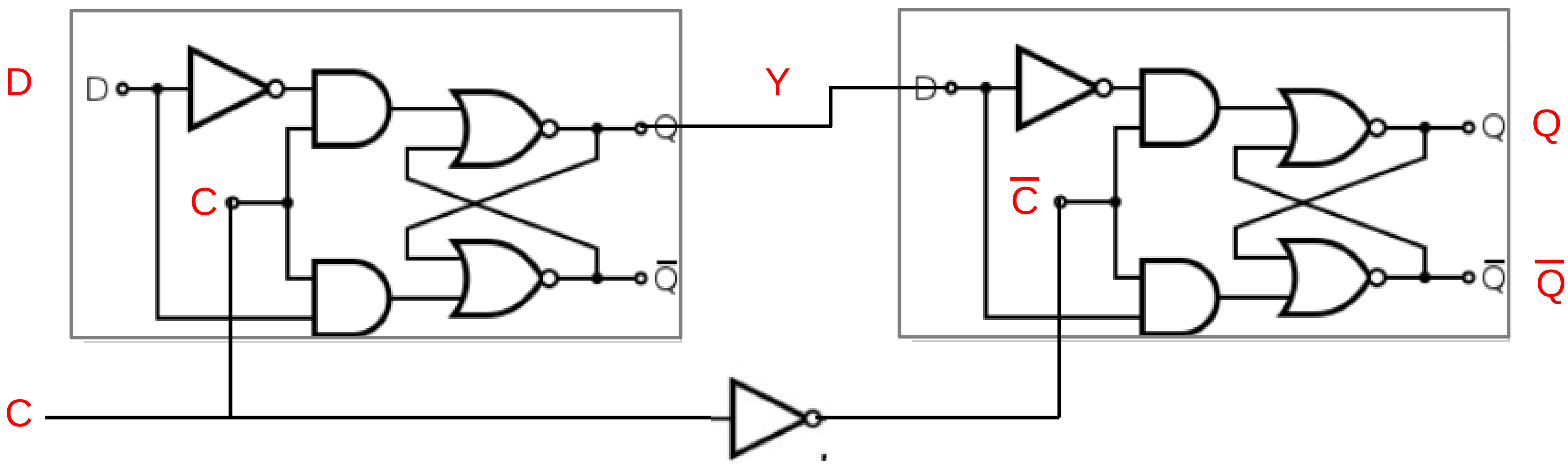


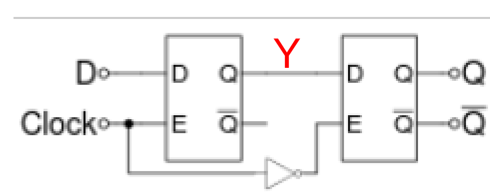


Trans 1	C=**1** D=**1**	Q=1 $\overline{Q}$=0
Trans 0	C=**1** D=0	Q=0 $\overline{Q}$=1
Opaque	C=**0** D=X	Q=old Q $\overline{Q}$=old $\overline{Q}$

https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Master-Slave FlipFlops

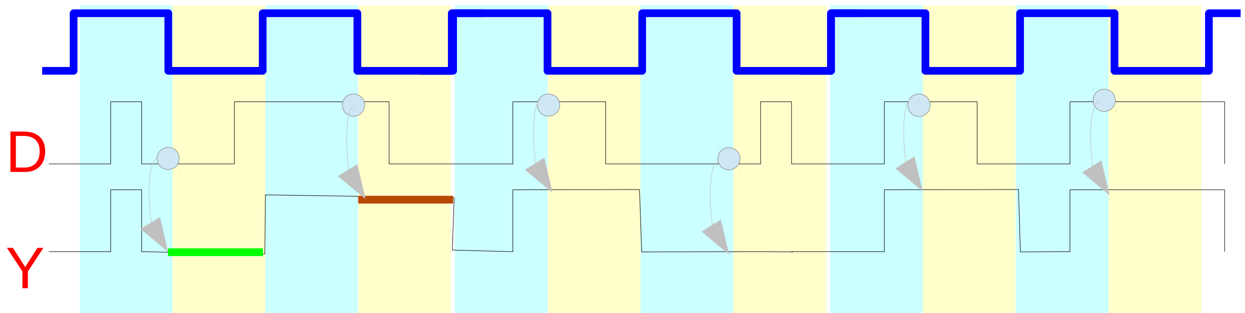


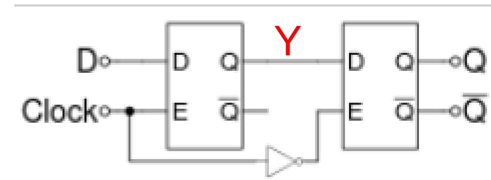


https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

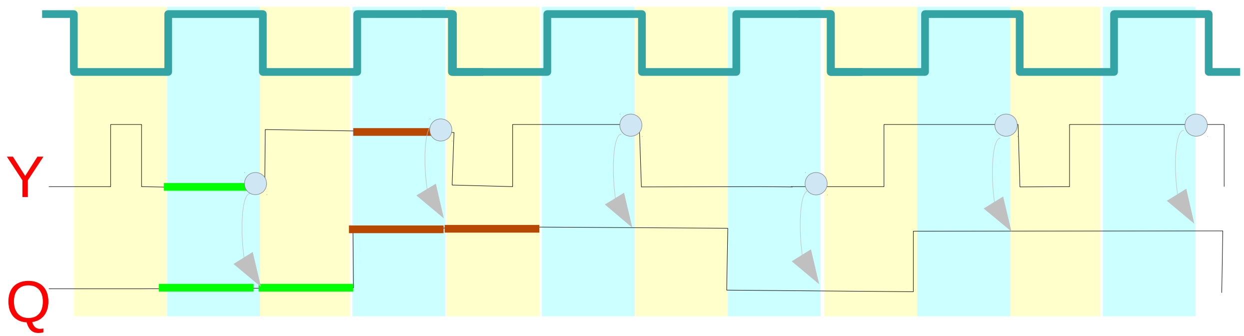
Master-Slave D FlipFlop

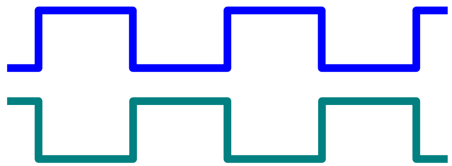
Master D Latch



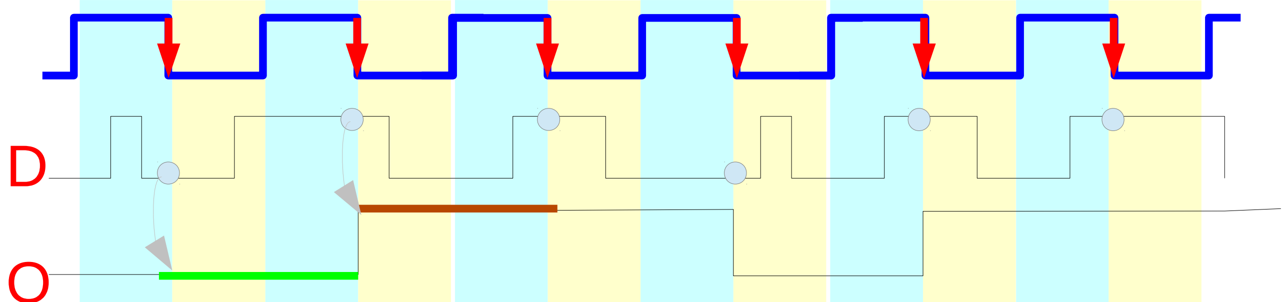


Slave D Latch

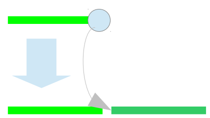




Master-Slave D F/F



https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

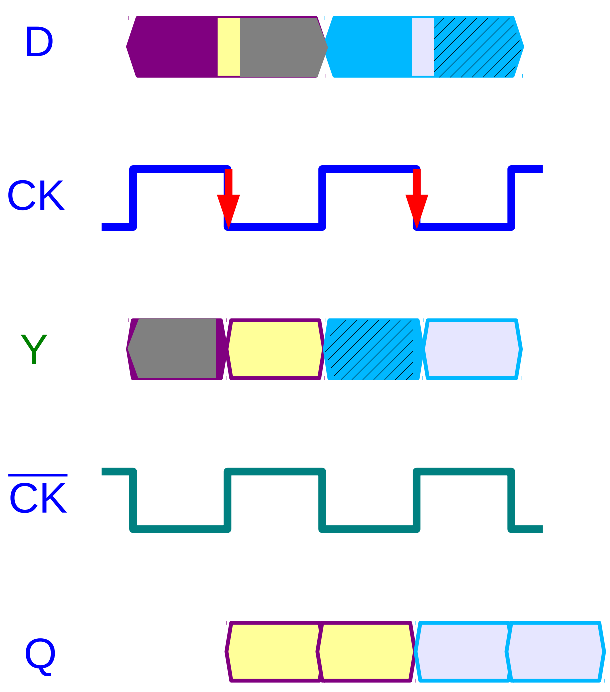


the hold output of the master is transparently reaches the output of the slave

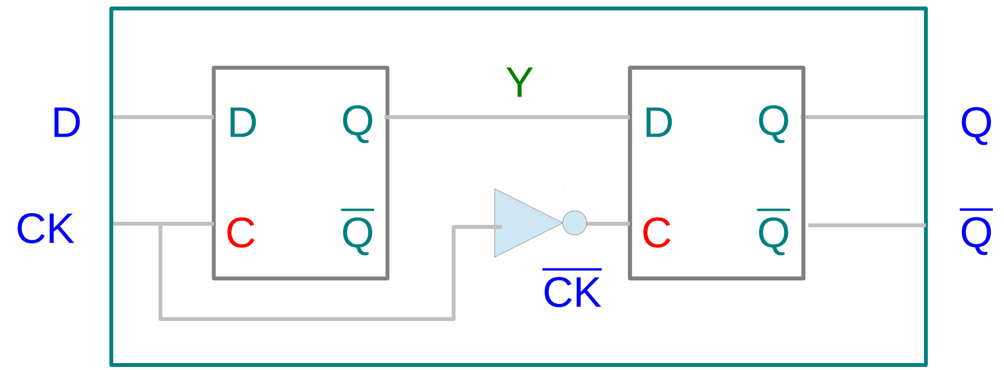
this value is held for another half period

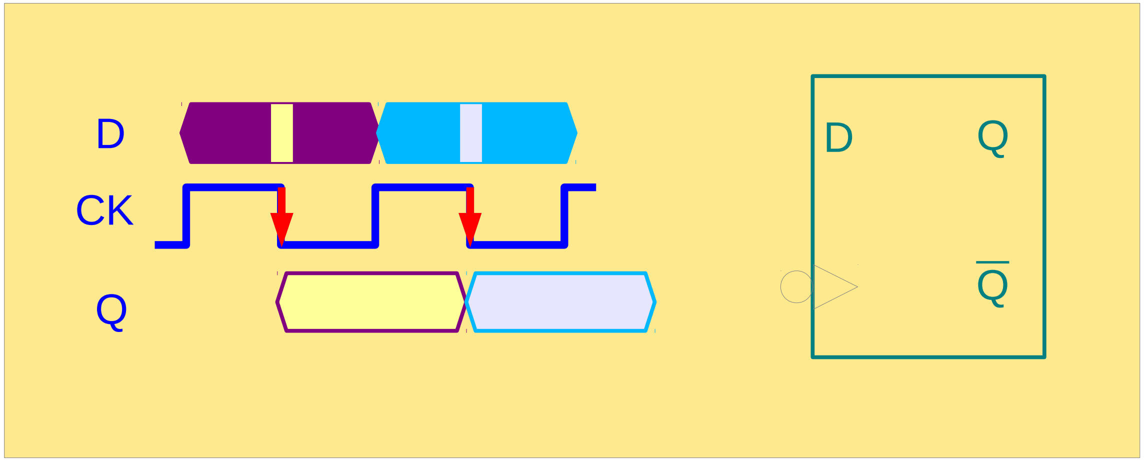
Master-Slave D FlipFlop – Falling Edge

Master D Latch



Slave D Latch



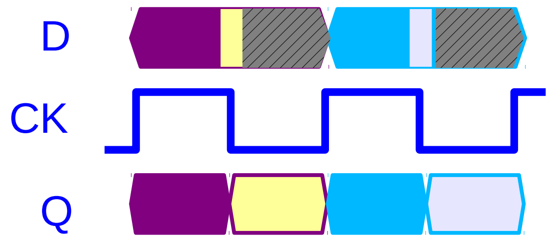


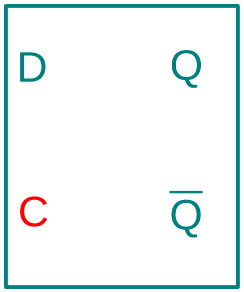
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

D Latch & D FlipFlop

Level Sensitive D Latch

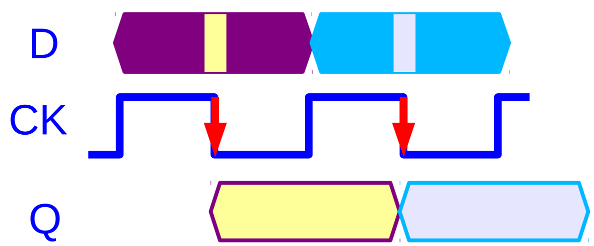
CK=1 transparent
CK=0 opaque

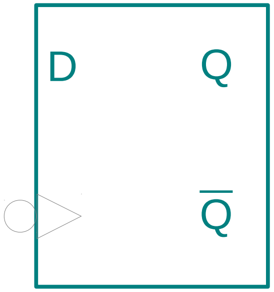




Edge Sensitive D FlipFlop

CK=1→0 transparent
else opaque

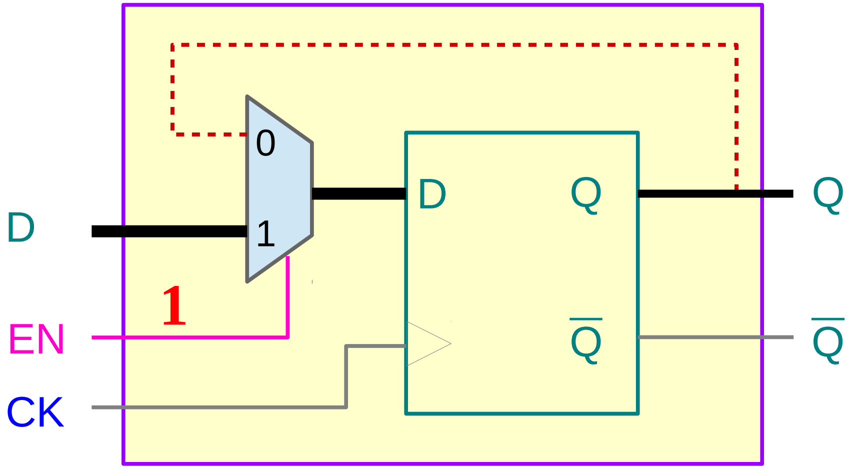




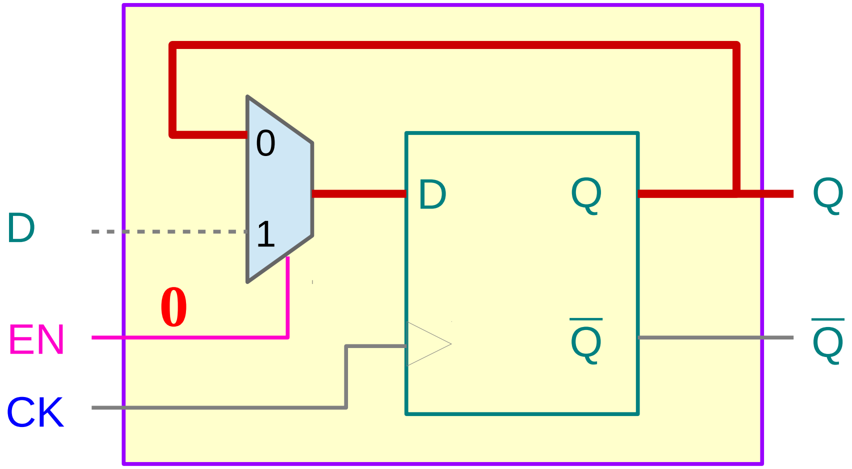
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

D FlipFlop with Enable (1)

EN=1 Regular D Flip Flop
Sampling **D** input @ **posedge** of CK

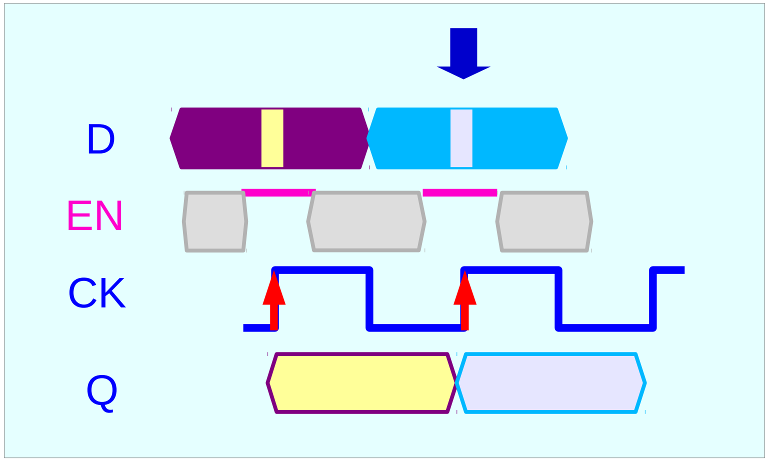


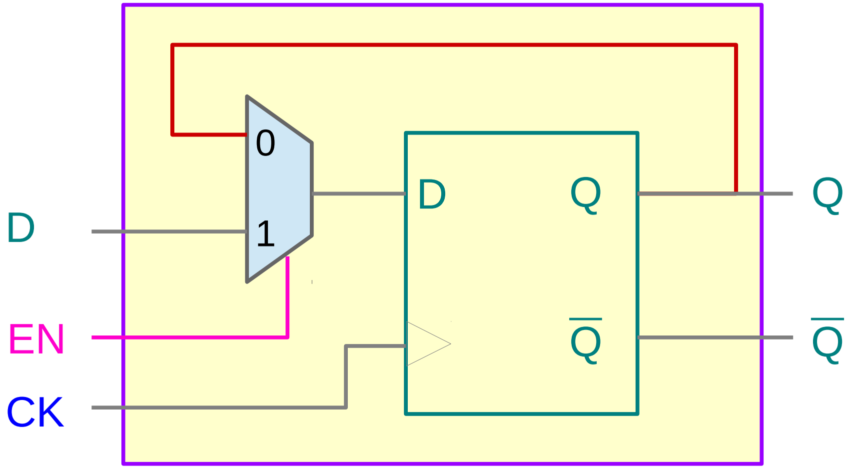
EN=0 Holding D Flip Flop
Sampling **Q** output @ **posedge** of CK

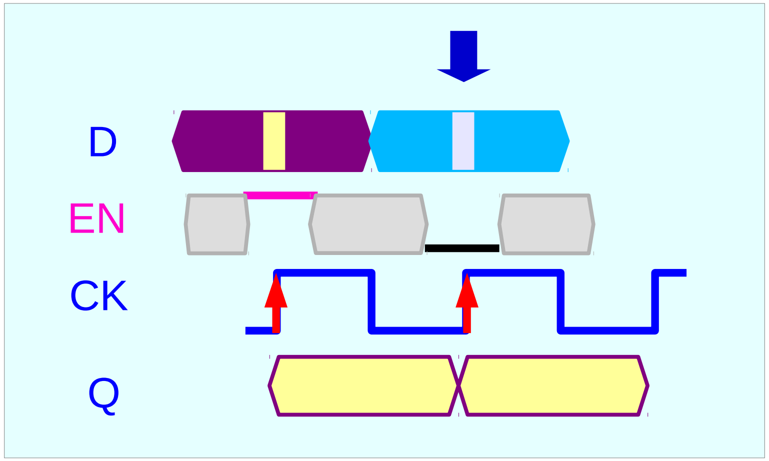


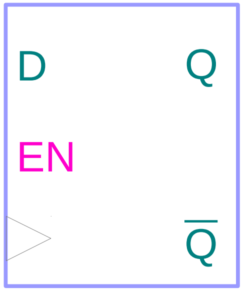
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

D FlipFlop with Enable (2)



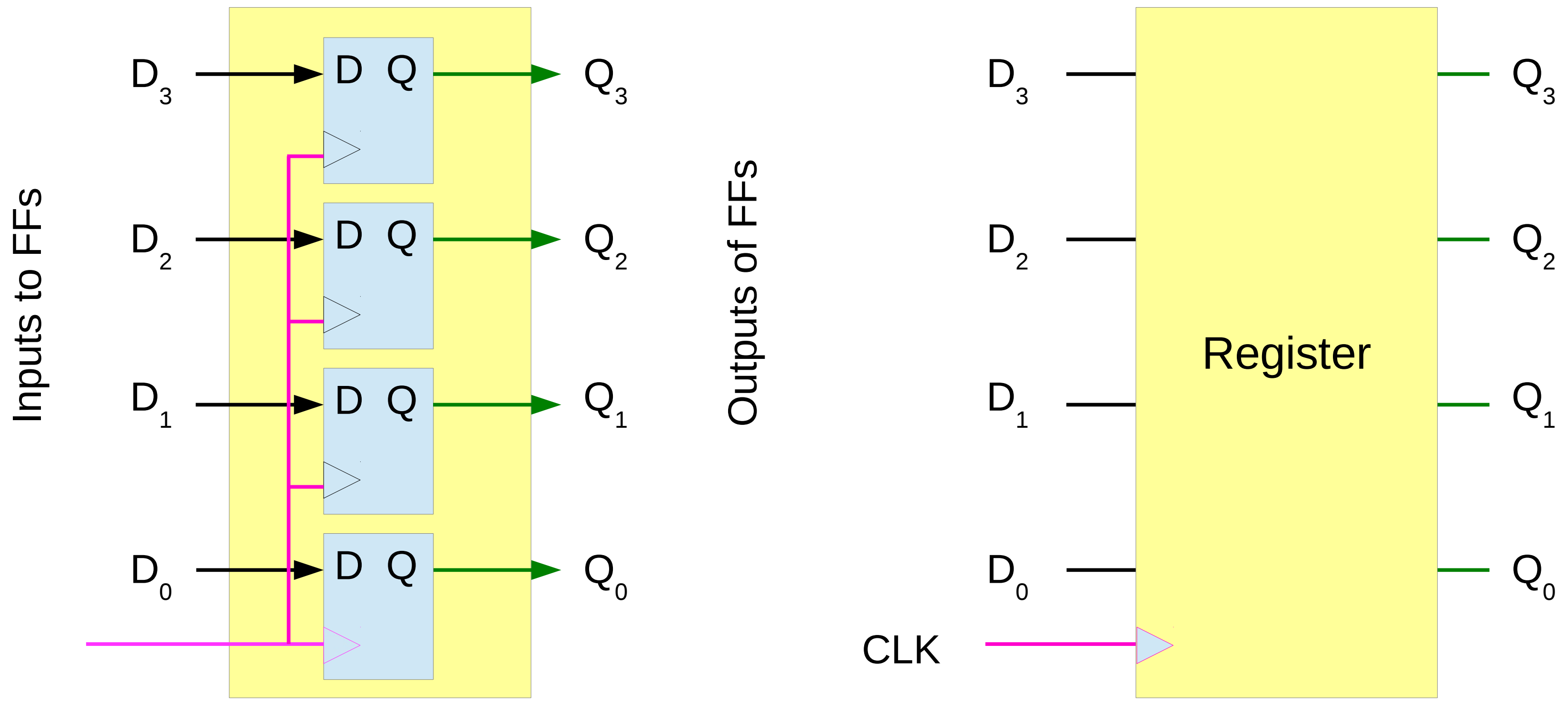






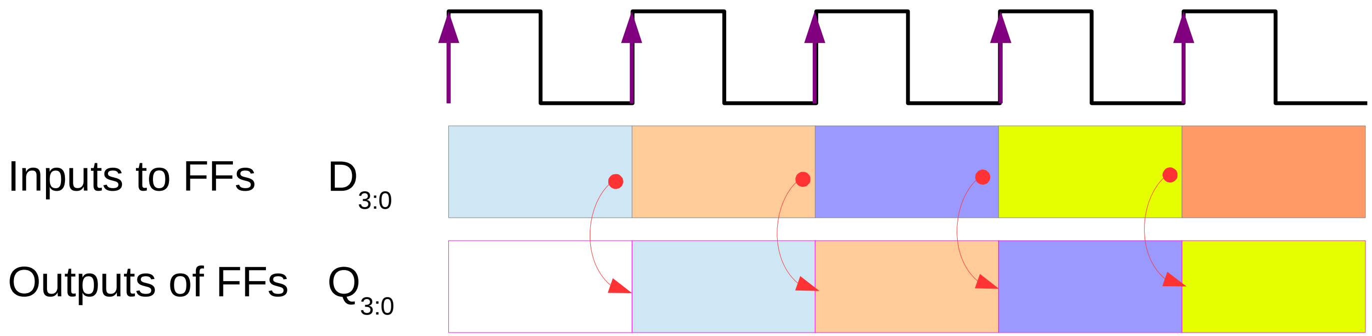
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

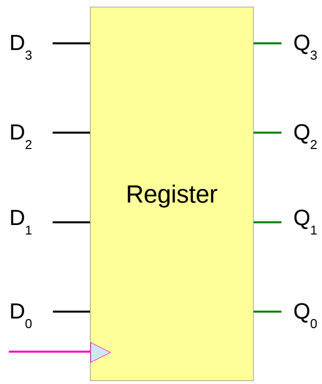
Registers



https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

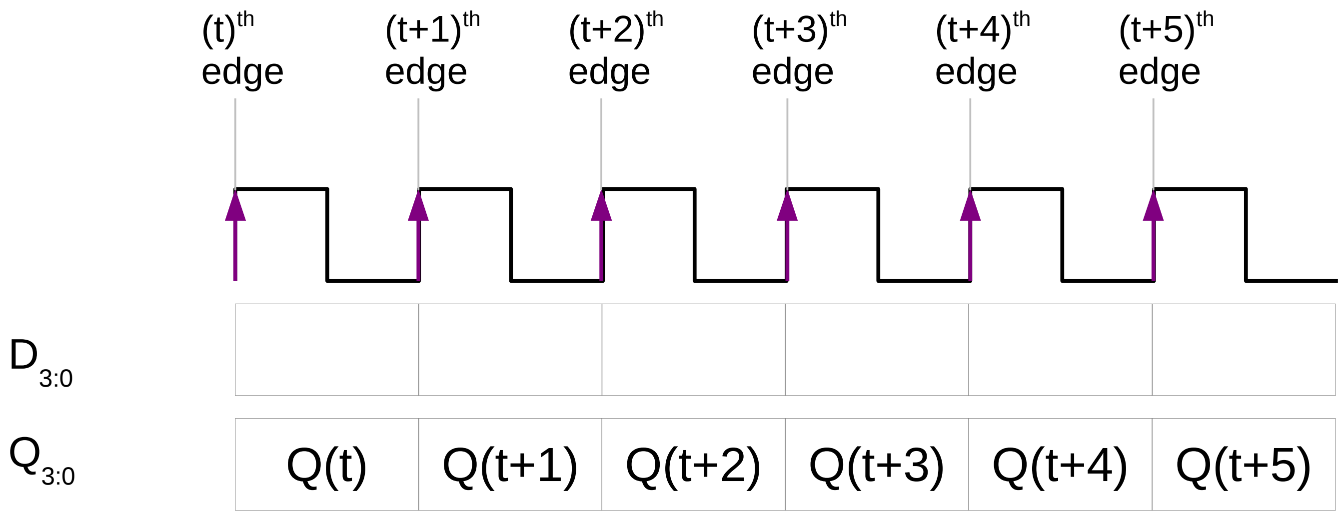
FF Timing (Ideal)

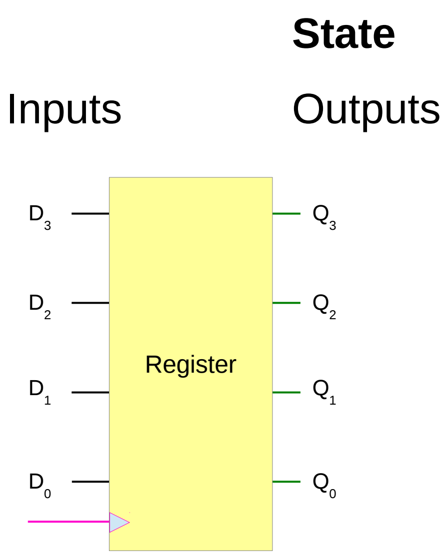




https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

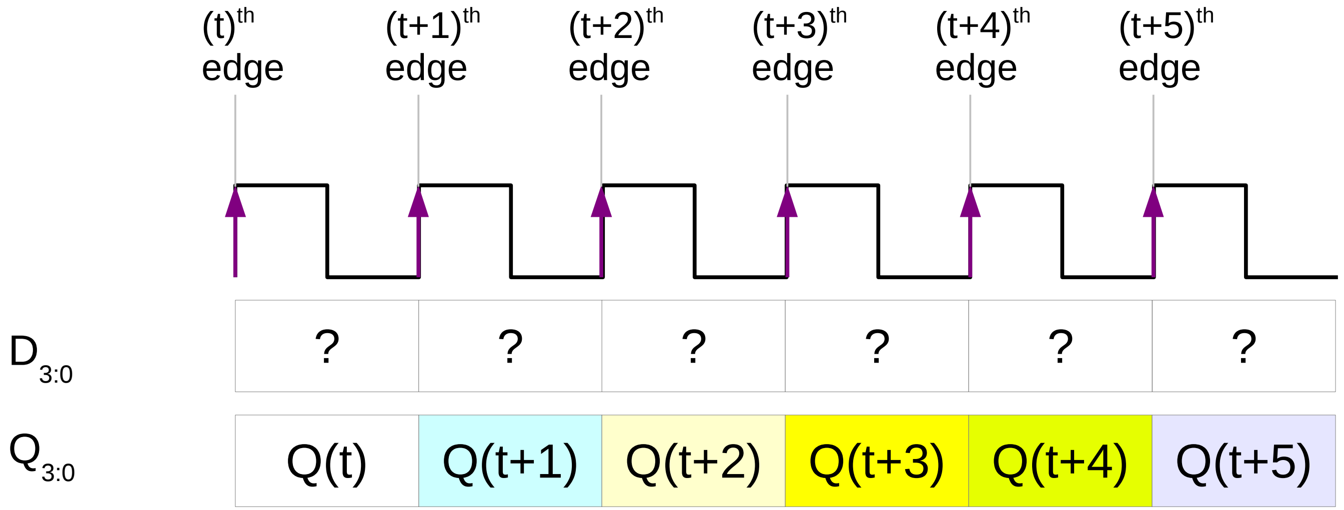
States

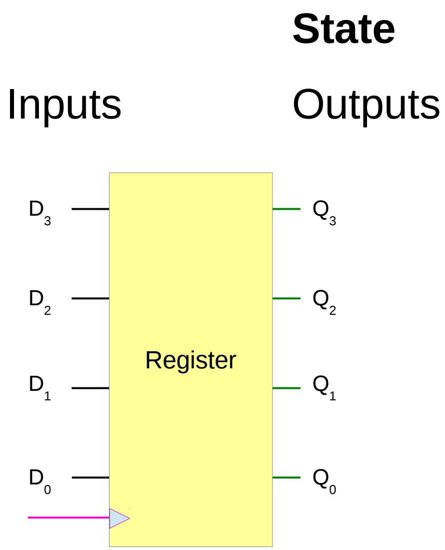




https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Sequence of States



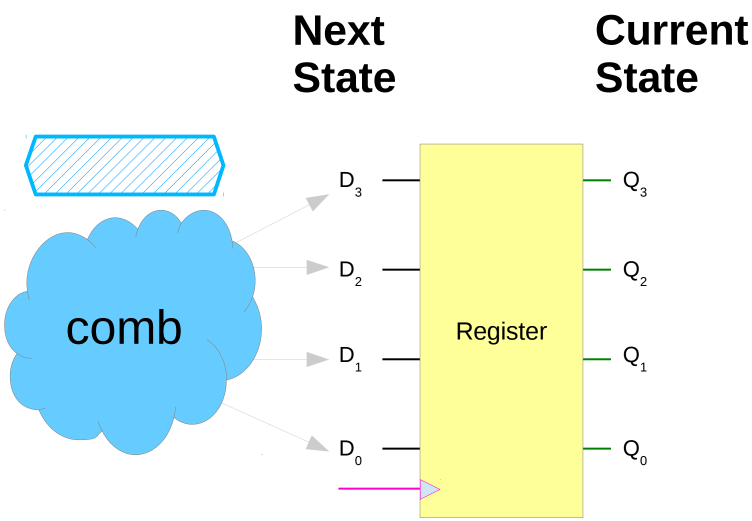


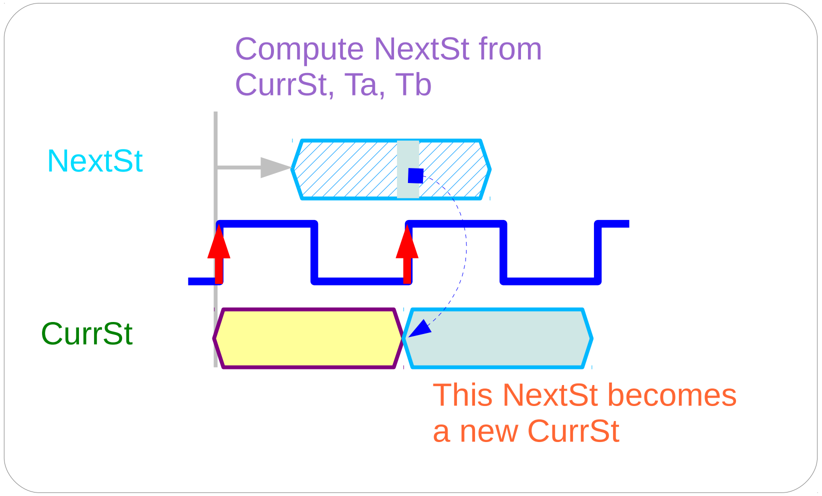
Find inputs to FFs

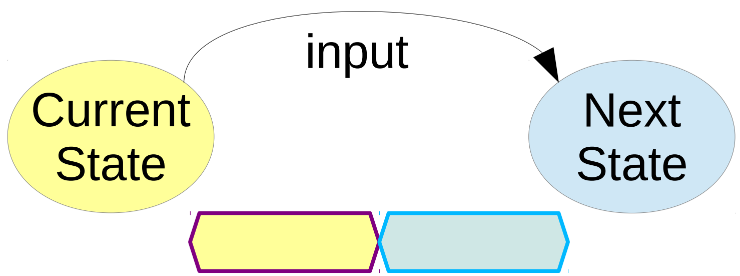
which will make outputs
in this sequence

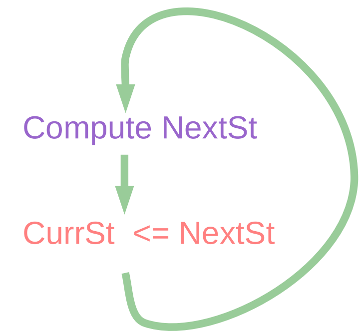
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

How to change current state



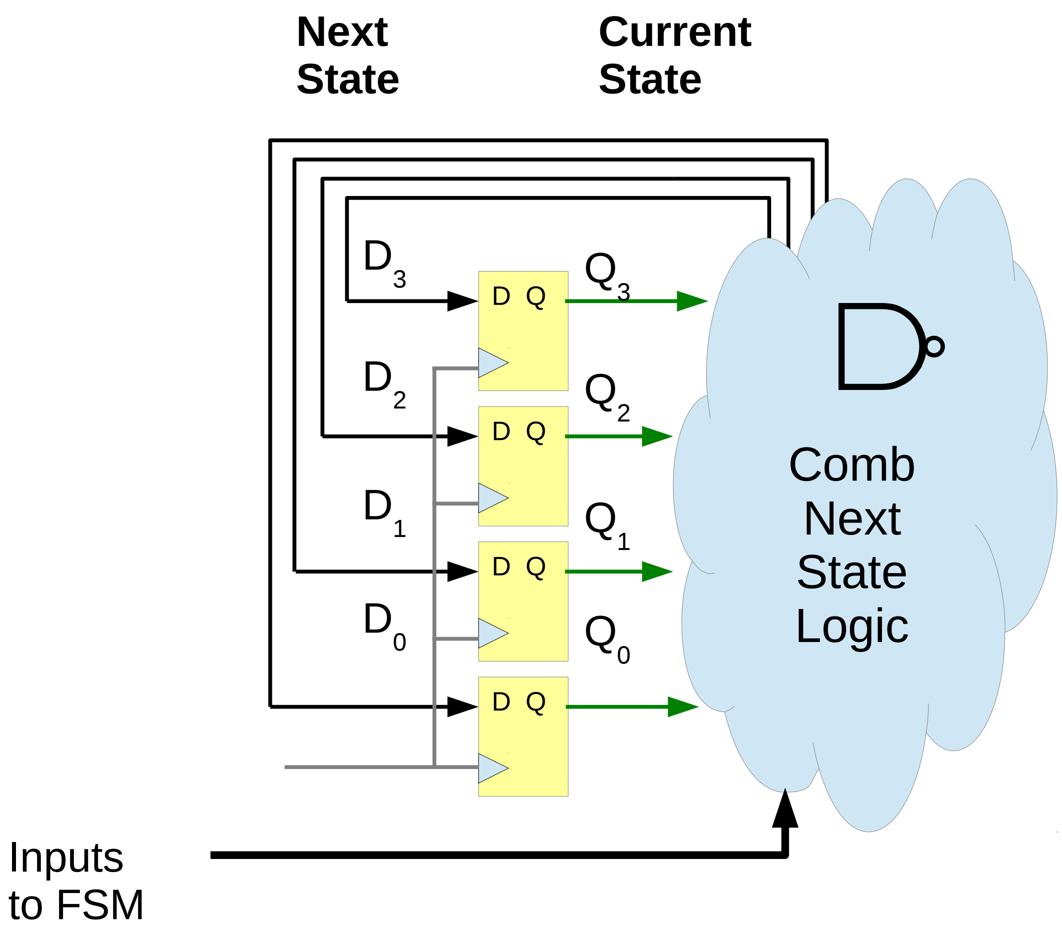






https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Finding FF Inputs



During the t^{th} clock edge period,

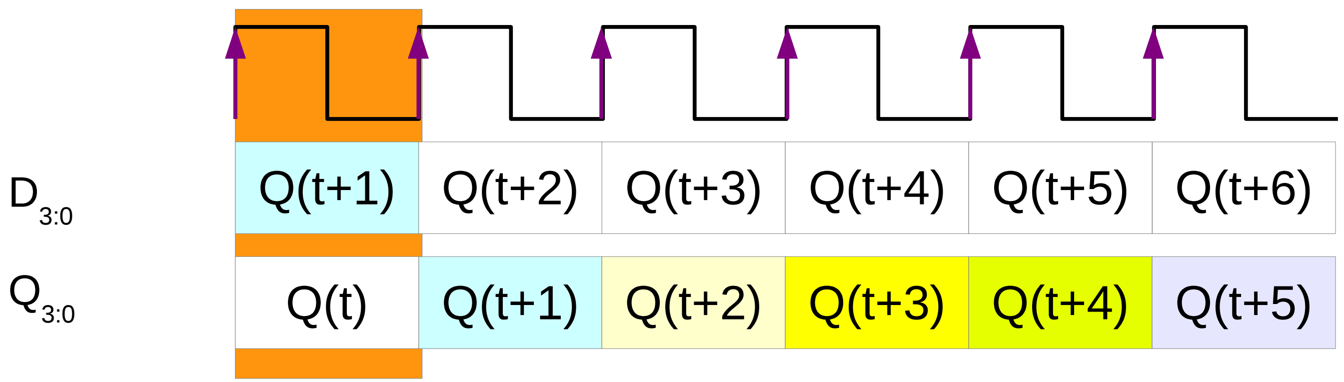
Compute the next state Q(t+1) using the current state Q(t) and other external inputs

Place it to FF inputs

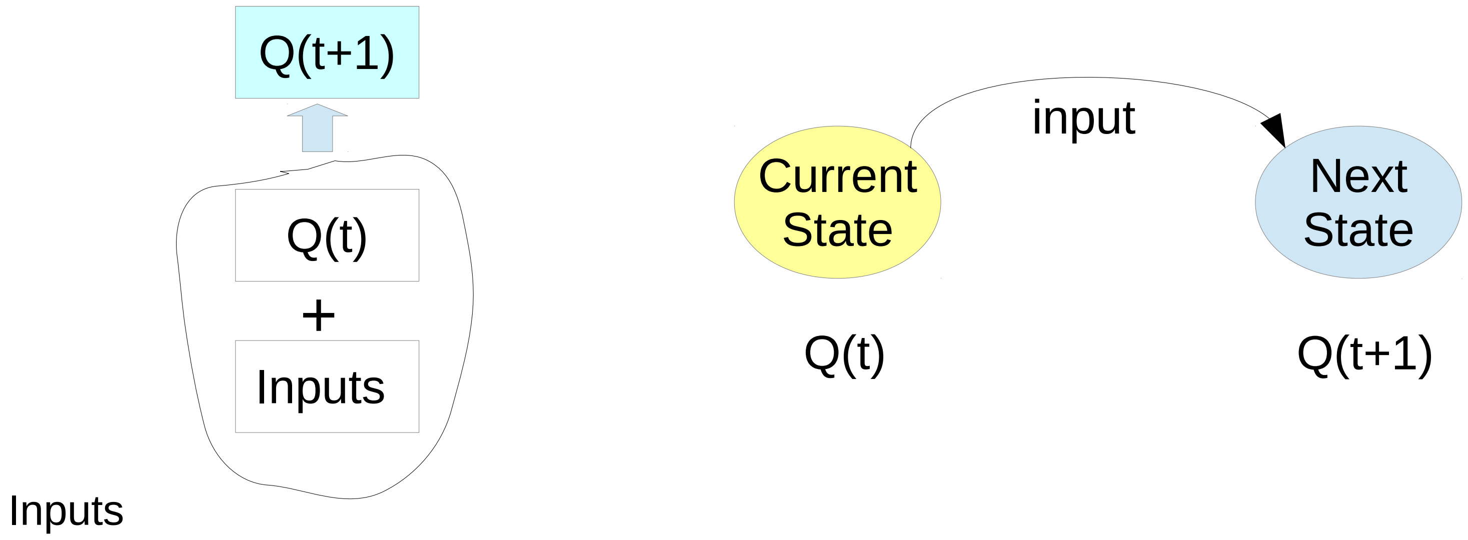
After the next clock edge, $(t+1)^{th}$, the computed next state Q(t+1) becomes the current state

https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

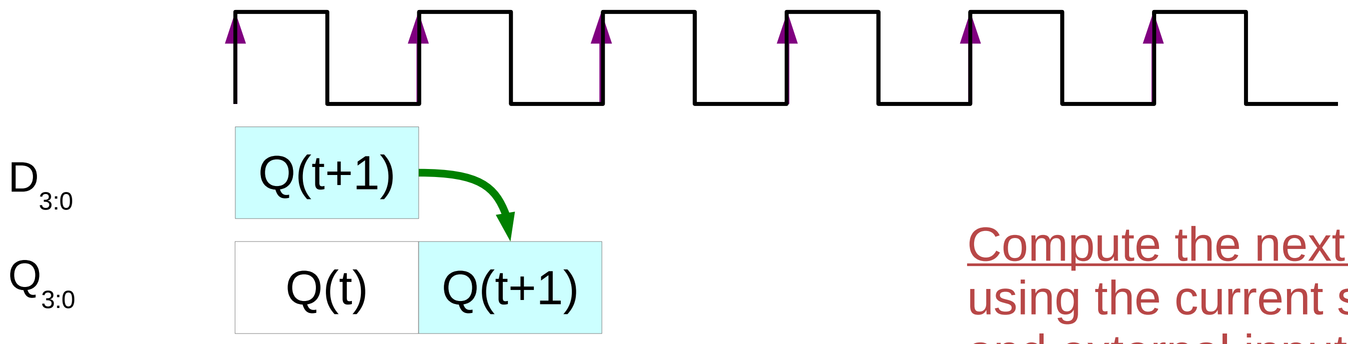
Method of Finding FF Inputs



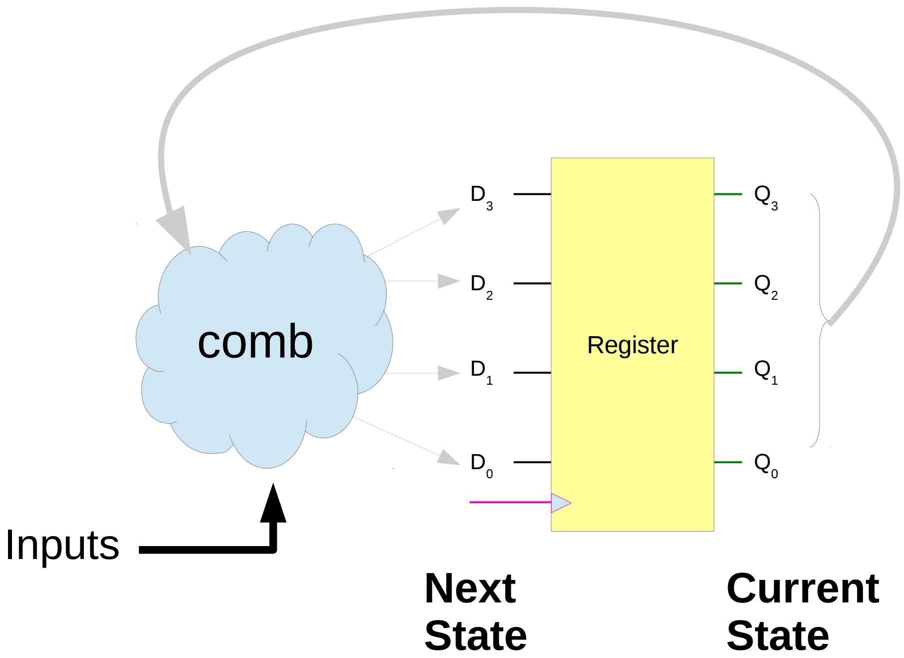
Find the boolean functions
D3, D2, D1, D0
in terms of Q3, Q2, Q1, Q0,
and external inputs
for all possible cases.

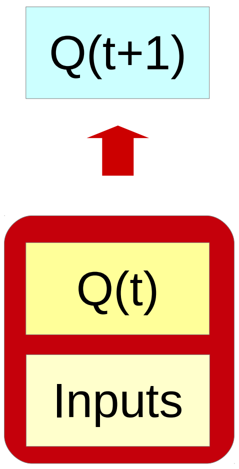


State Transition



Compute the next state
using the current state
and external inputs
in the current clock cycle

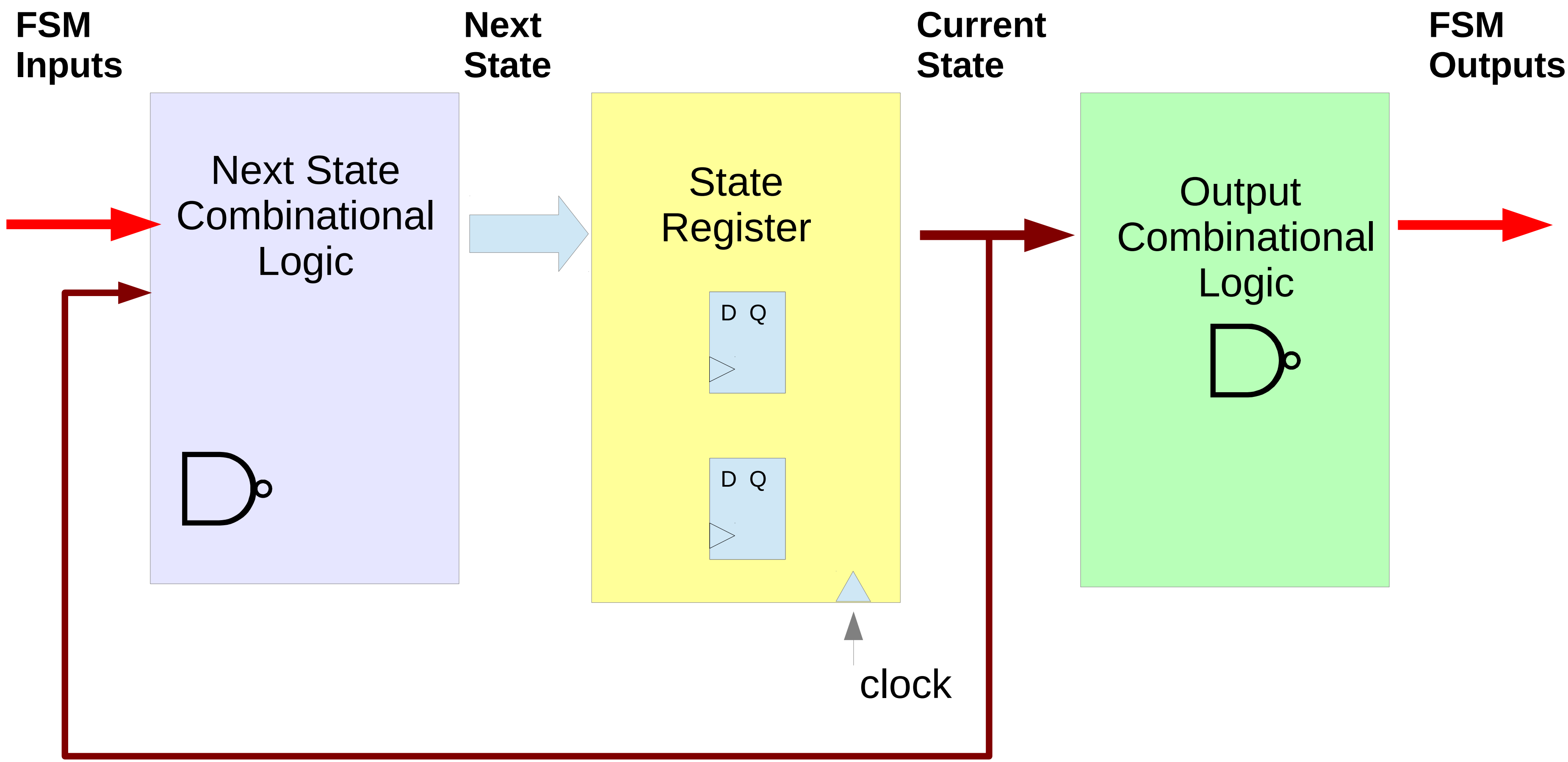




After the next clock edge,
the computed next state (FF Inputs)
becomes the current state (FF Outputs)

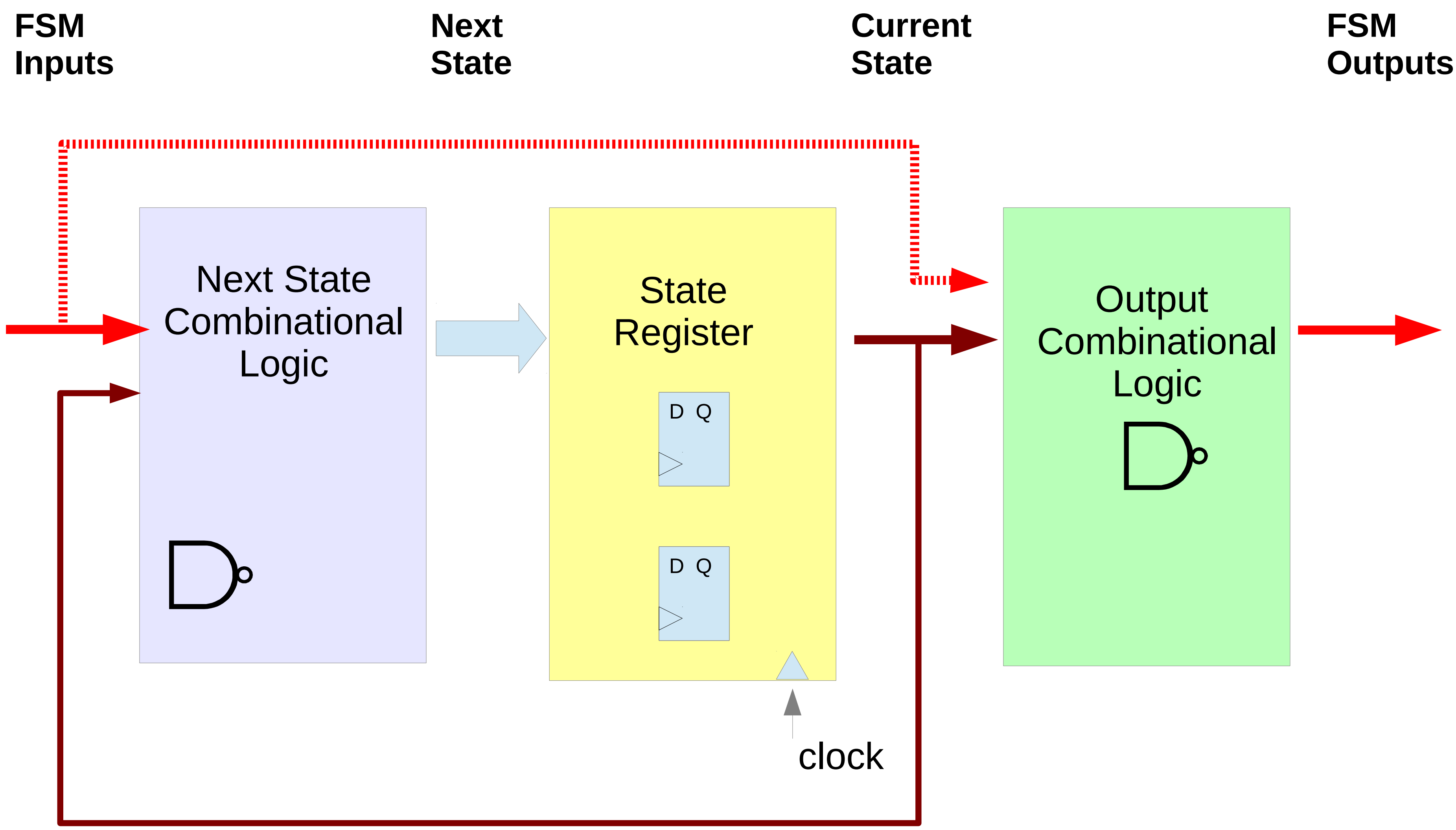
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Moore FSM



https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Mealy FSM

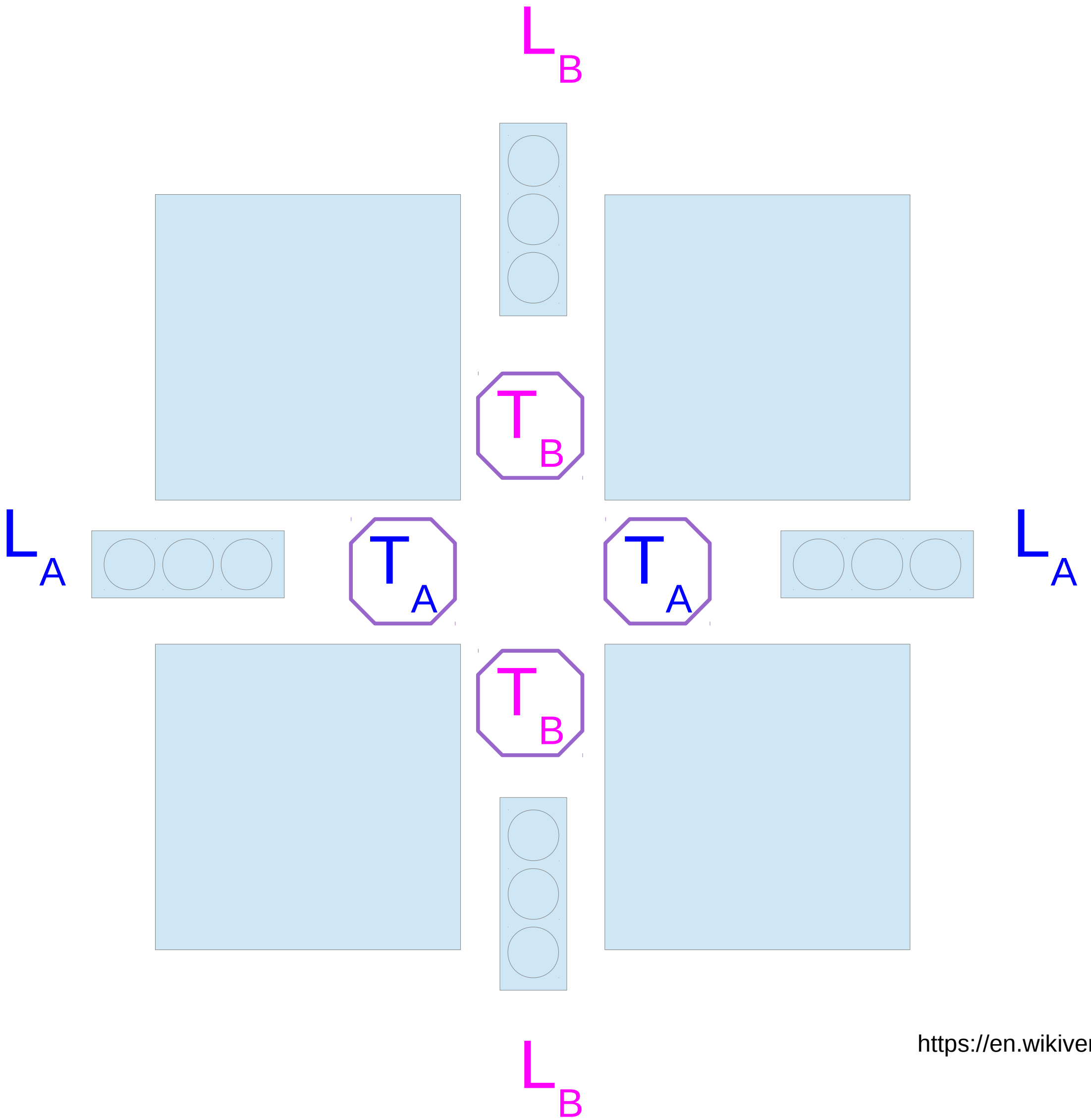


https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Traffic Lights Example

https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

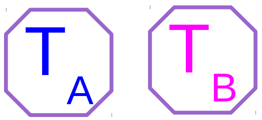
FSM Inputs and Outputs



Traffic Lights - Outputs

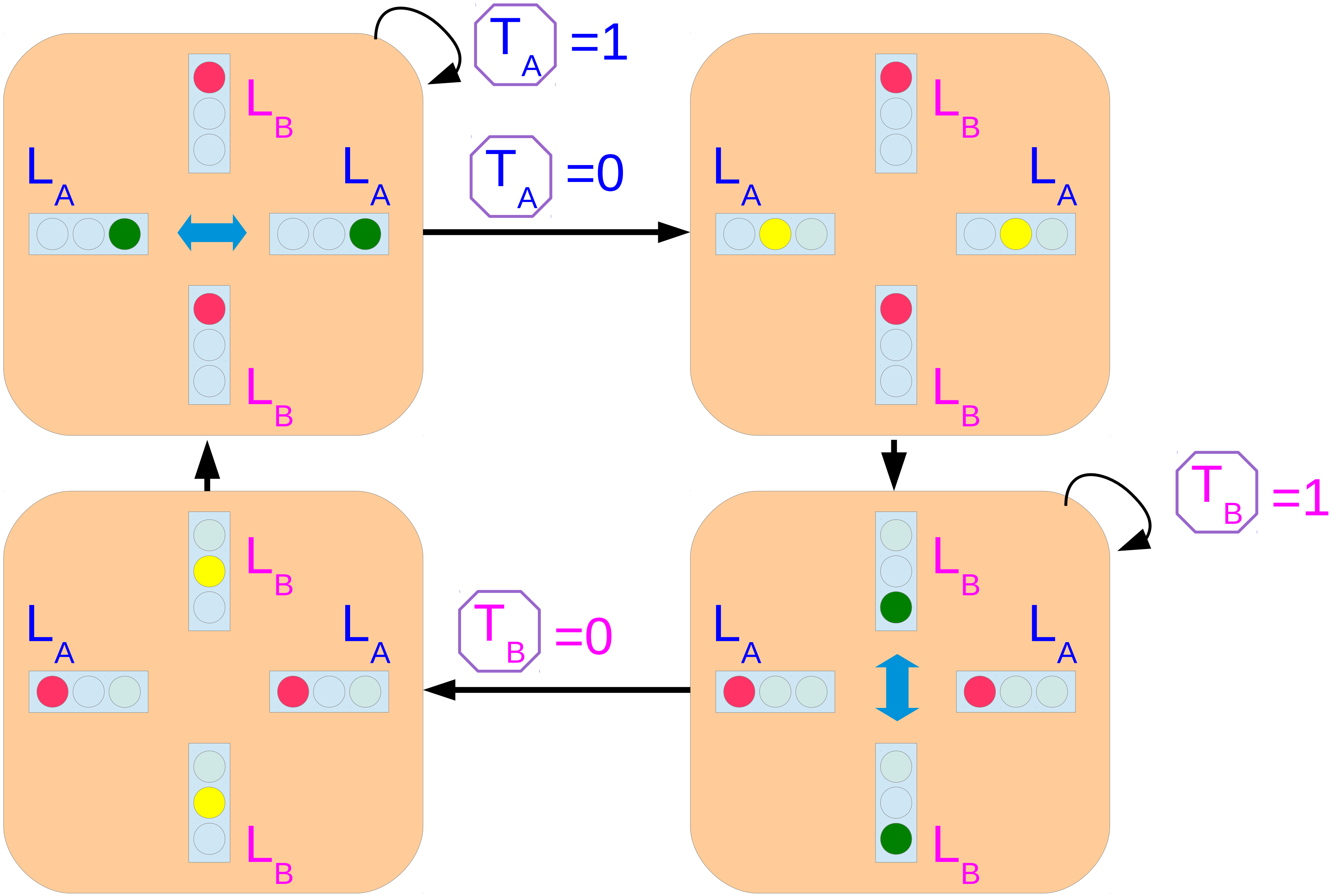
L_A L_B

Sensor - Inputs



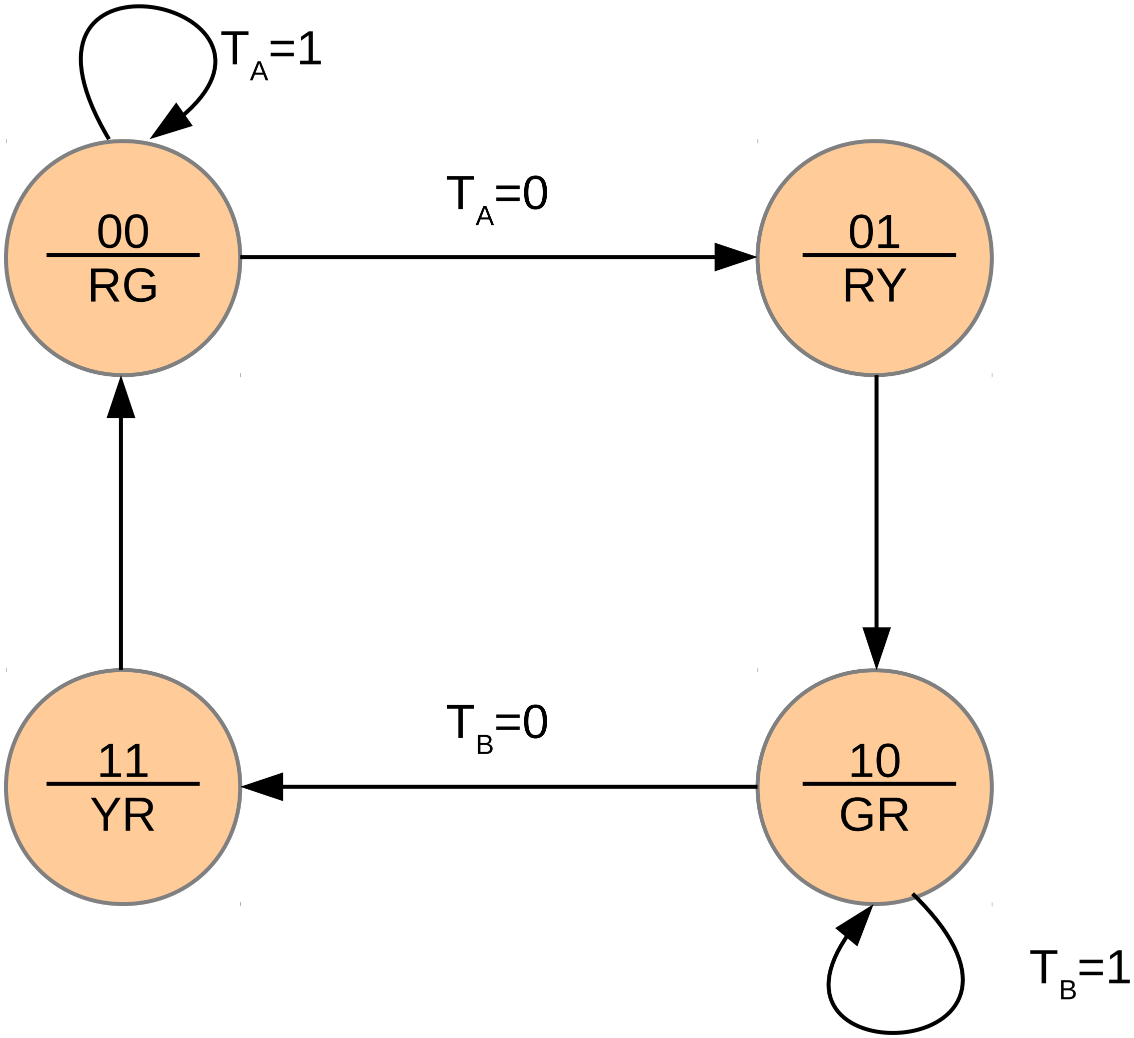
https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

States



https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

State and State Transition Diagrams



S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

S_1	S_2	L_{A1}	L_{A0}	L_{B1}	L_{B0}		
0	0	0	0	1	0	R	G
0	1	0	1	1	0	Y	G
1	0	1	0	0	0	G	R
1	1	1	0	0	1	G	0

Next State Functions S_1’ and S_2’

S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

	S_1	S_0	T_A	T_B	S'_1
	0	0	0	X	0
	0	0	1	X	0
$\overline{S_1} S_0$ ⇨	0	1	X	X	1
$S_1 \overline{S_0} \overline{T_B}$ ⇨	1	0	X	0	1
$S_1 \overline{S_0} T_B$ ⇨	1	0	X	1	1
	1	1	X	X	0

$$S'_1 = \overline{S_1} S_0 + S_1 \overline{S_0}$$
$$= S_1 \oplus S_0$$

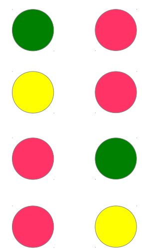
	S_1	S_0	T_A	T_B	S'_0
$\overline{S_1} \overline{S_0} \overline{T_A}$ ⇨	0	0	0	X	1
	0	0	1	X	0
	0	1	X	X	0
$S_1 \overline{S_0} \overline{T_B}$ ⇨	1	0	X	0	1
	1	0	X	1	0
	1	1	X	X	0

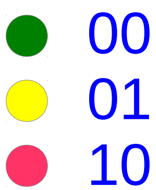
$$S'_0 = \overline{S_1} \overline{S_0} \overline{T_A} + S_1 \overline{S_0} \overline{T_B}$$

https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

Output Functions : L_{A1}, L_{A0}, L_{B0}, L_{B1}

S_1	S_2	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1





S_1	S_2	L_{A1}
0	0	0
0	1	0
1	0	1
1	1	1

$$L_{A1} = S_1$$

S_1	S_2	L_{A0}
0	0	0
0	1	1
1	0	0
1	1	0

$$L_{A0} = \overline{S_1} S_0$$

S_1	S_2	L_{B1}
0	0	1
0	1	1
1	0	0
1	1	0

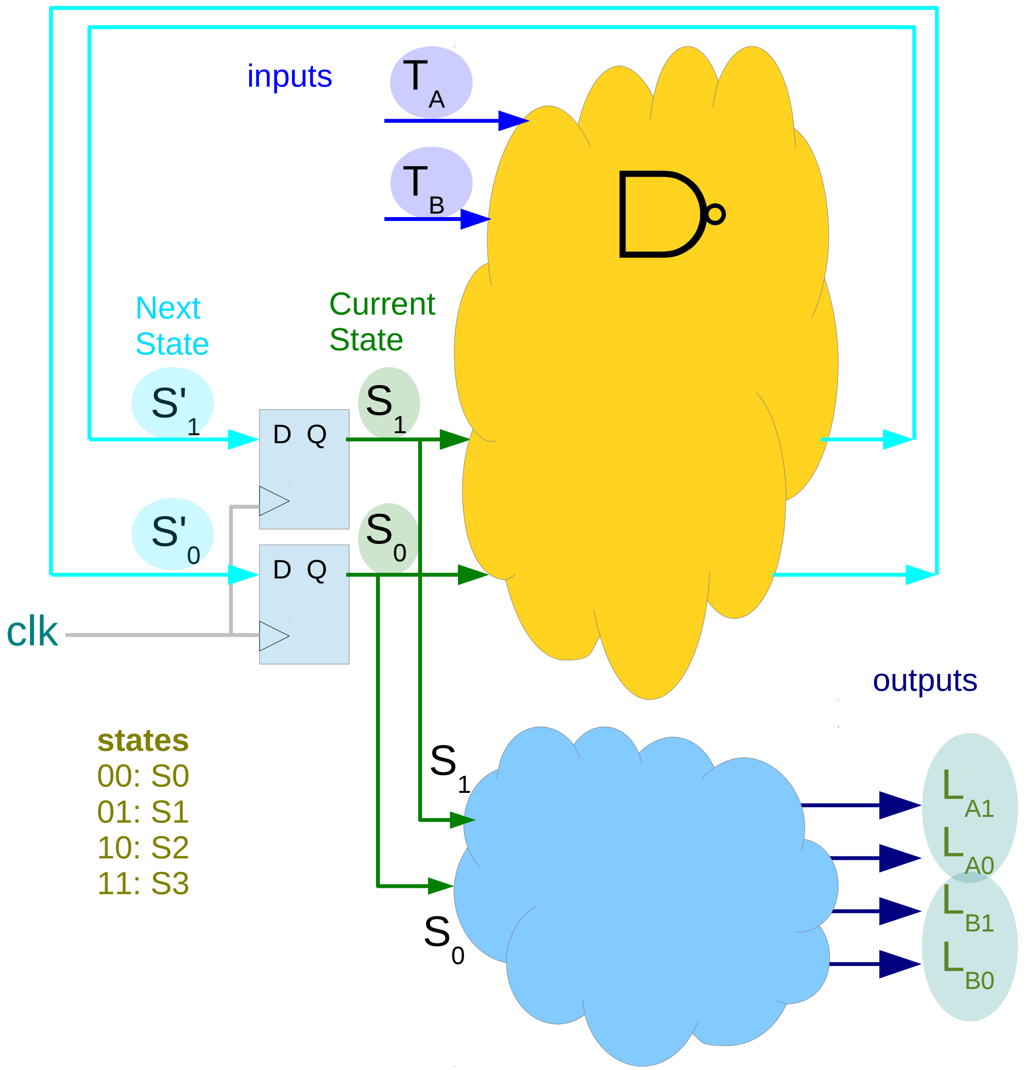
$$L_{B1} = \overline{S_1}$$

S_1	S_2	L_{B0}
0	0	0
0	1	0
1	0	0
1	1	1

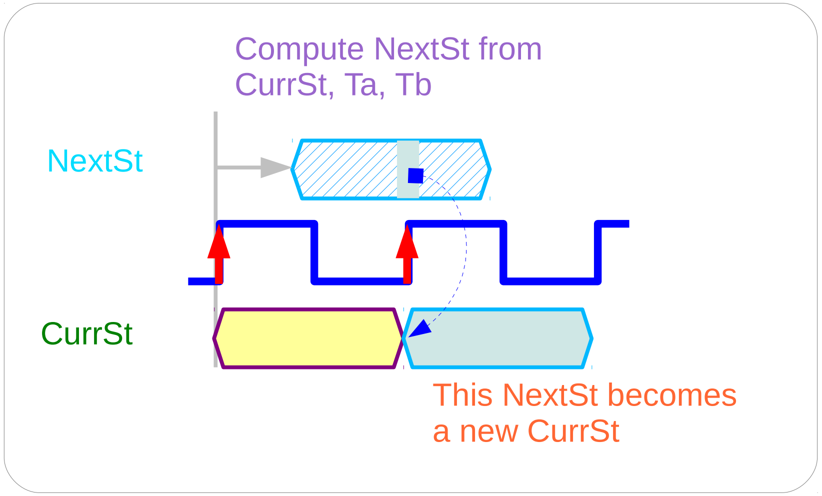
$$L_{A0} = S_1 S_0$$

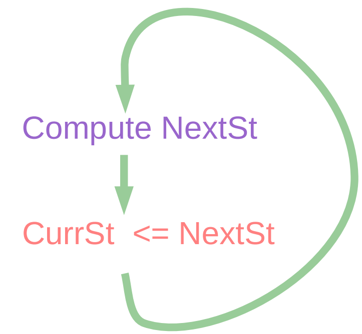
https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

Moore FSM



states
00: S0
01: S1
10: S2
11: S3

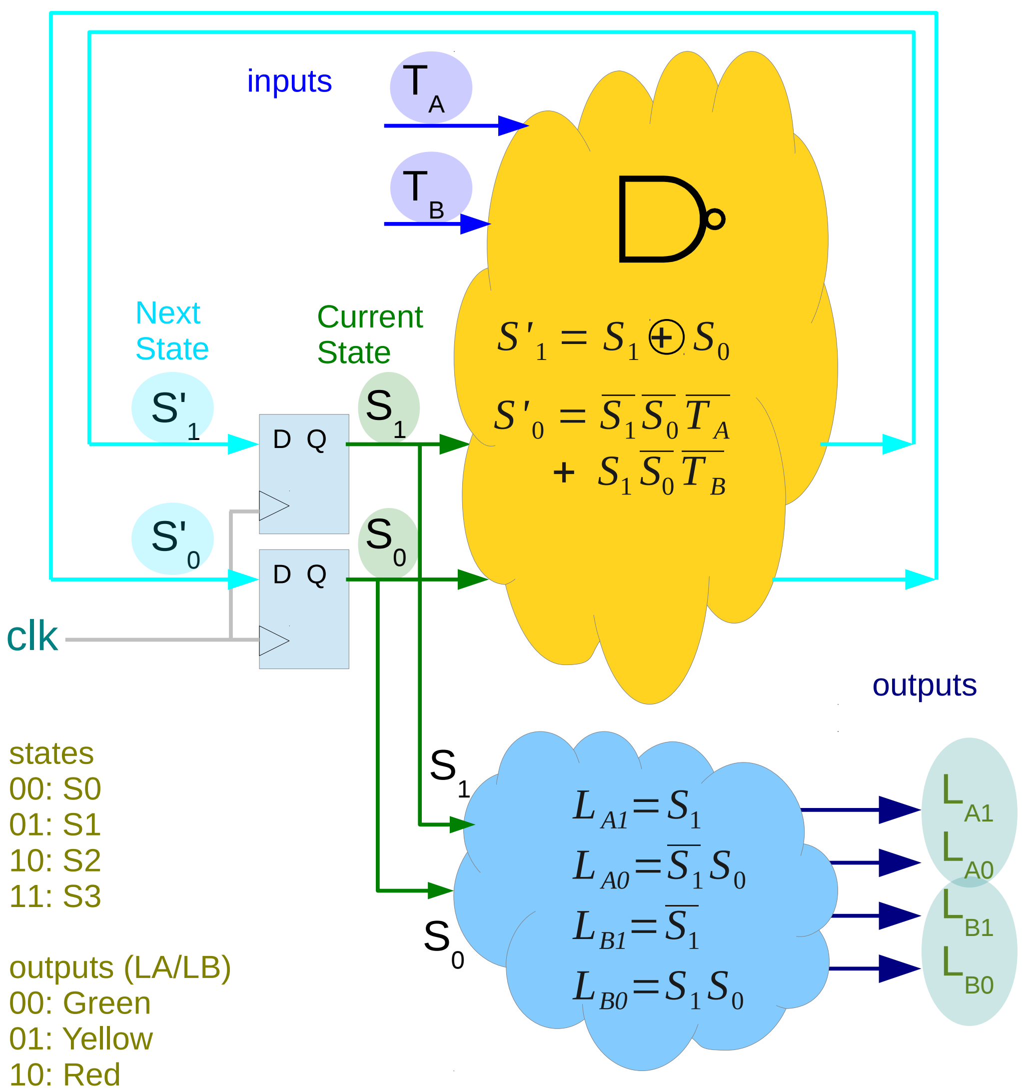




outputs (LA/LB)
00: Green
01: Yellow
10: Red
11: X

https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

Moore FSM Implementation



states
00: S0
01: S1
10: S2
11: S3

outputs (LA/LB)
00: Green
01: Yellow
10: Red
11: X

Inputs T_A T_B

Current State S_1 S_0

Next States

$$S'_1 = S_1 \oplus S_0$$

$$S'_0 = \overline{S_1}\,\overline{S_0}\,\overline{T_A} + S_1\overline{S_0}\,\overline{T_B}$$

Current State S_1 S_0

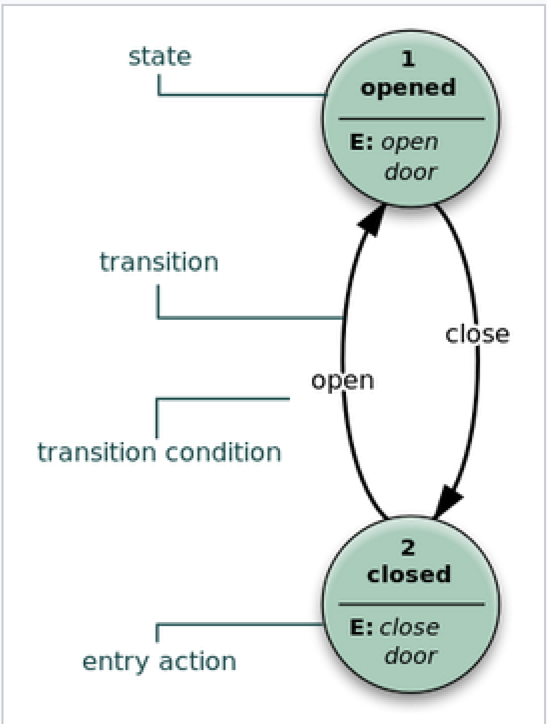
Outputs

$L_{A1} = S_1$ $L_{B1} = \overline{S_1}$

$L_{A0} = \overline{S_1} S_0$ $L_{B0} = S_1 S_0$

https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

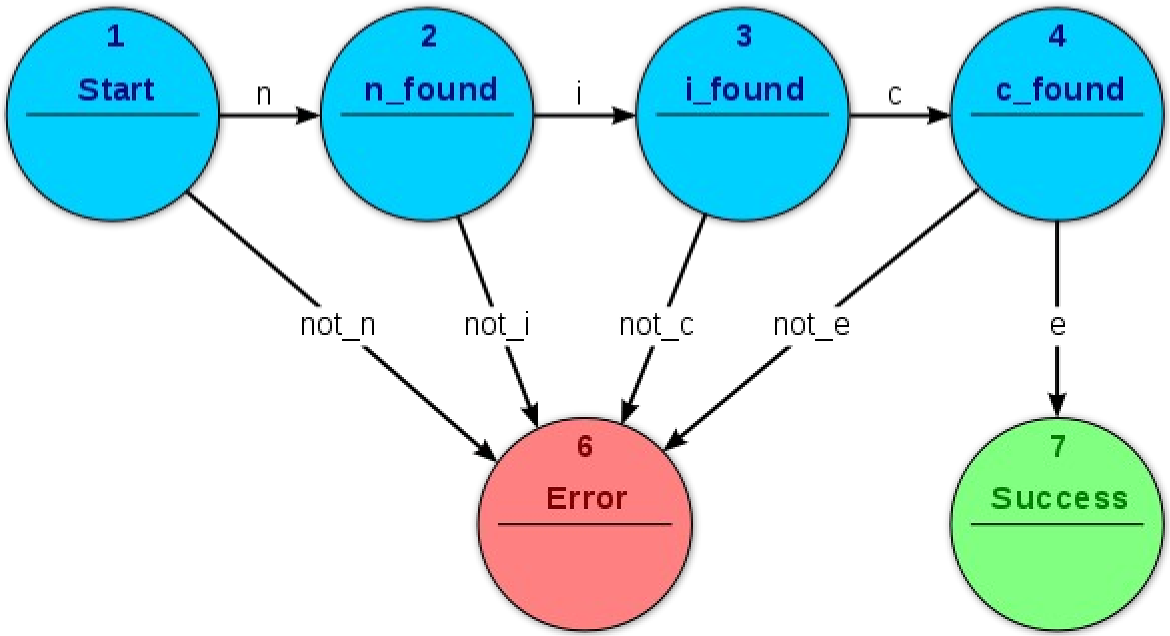
State Diagram



A state diagram for a door that can only be opened and closed

https://en.wikipedia.org/wiki/Finite-state_machine

Acceptors and Recognizers



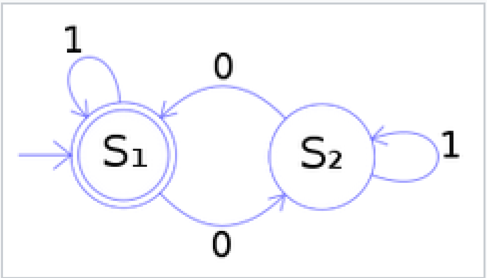


Fig. 5: Representation of a finite-state machine; this example shows one that determines whether a binary number has an even number of 0s, where S_1 is an **accepting state**.

Acceptor FSM: parsing the string "nice"

https://en.wikipedia.org/wiki/Finite-state_machine

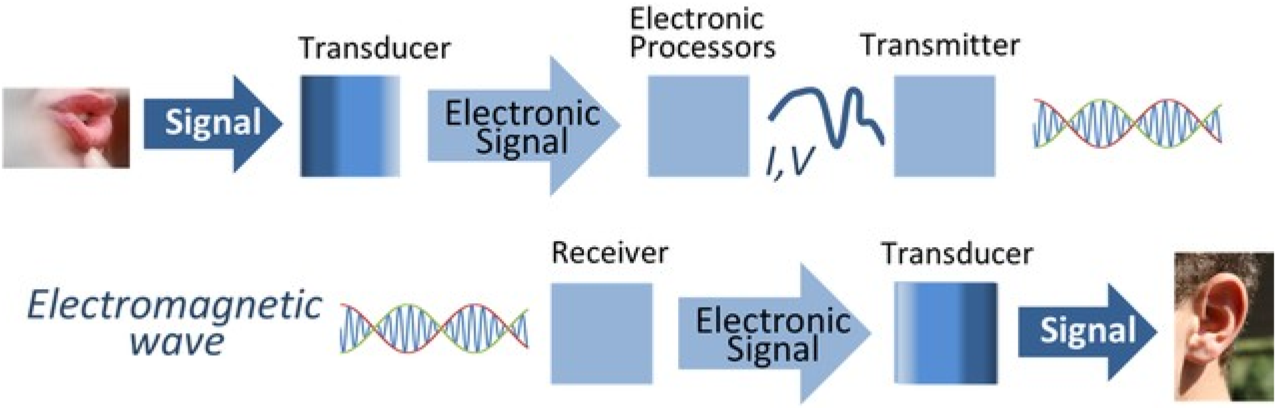
Classifiers and Transducers

A **classifier** is a generalization of
a finite state machine that,
similar to an acceptor,
produces a <u>single</u> <u>output</u> on <u>termination</u>
but has <u>more than two</u> **terminal states**

Transducers generate **output** based on a given **input** and/or a **state** using actions.
They are used for <u>control</u> <u>applications</u> and in the field of computational linguistics.

https://en.wikipedia.org/wiki/Finite-state_machine

General Transducers



Transducers are used in electronic communications systems to convert signals of various physical forms to electronic signals, and vice versa. In this example, the first transducer could be a **microphone**, and the second transducer could be a **speaker**.

https://en.wikipedia.org/wiki/Transducer

Transducers : Moore and Mealy Machines

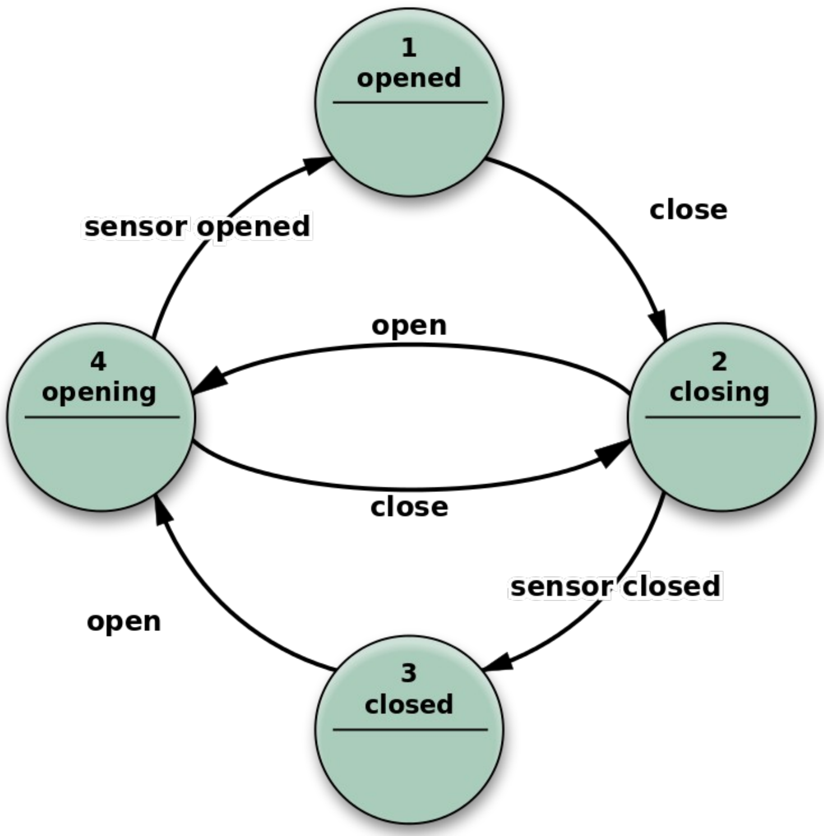


Fig. 6 Transducer FSM: Moore model example

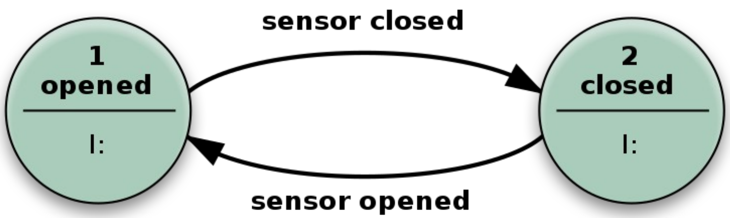


Fig. 7 Transducer FSM: Mealy model example

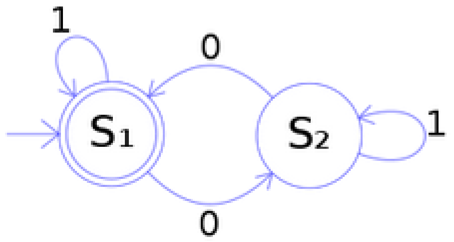
There are two **input actions** (I:): "start motor to close the door if command_close arrives" and "start motor in the other direction to open the door if command_open arrives".

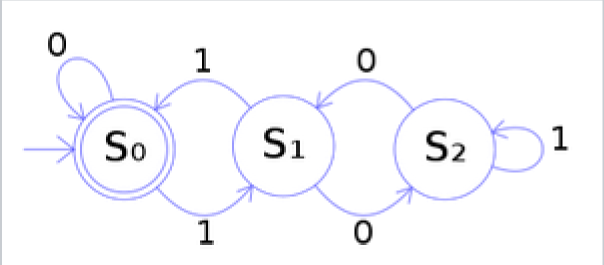
https://en.wikipedia.org/wiki/Finite-state_machine

Moore machine

Example: DFA, NFA, GNFA, or Moore machine [edit]

S_1 and S_2 are states and S_1 is an **accepting state** or a **final state**. Each edge is labeled with the input. This example shows an acceptor for strings over {0,1} that contain an even number of zeros.





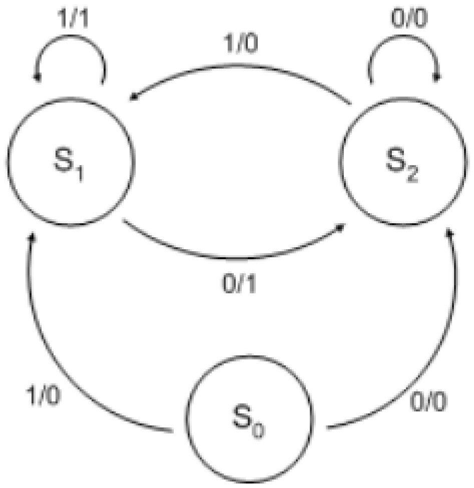
An example of a deterministic finite automaton that accepts only binary numbers that are multiples of 3. The state S_0 is both the start state and an accept state.

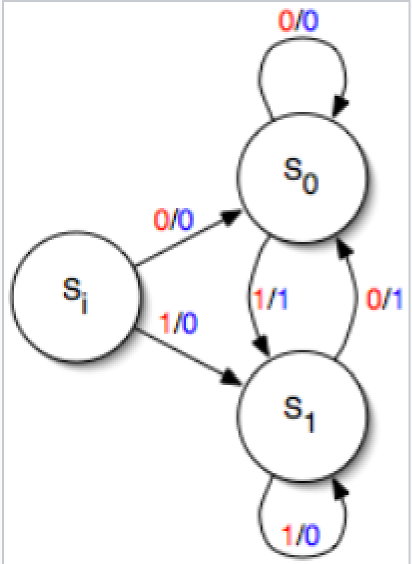
https://en.wikipedia.org/wiki/State_diagram
https://en.wikipedia.org/wiki/Finite-state_transducer

Mealy machine

Example: Mealy machine [edit]

S_0, S_1, and S_2 are states. Each edge is labeled with "*j* / *k*" where *j* is the input and *k* is the output.





State diagram for a simple Mealy machine with one input and one output.

https://en.wikipedia.org/wiki/State_diagram

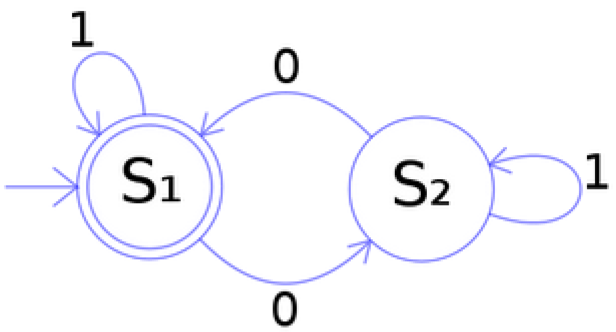
https://en.wikipedia.org/wiki/Mealy_machine

State Transition Table

State Transition Table

Input / State	1	0
S_1	S_1	S_2
S_2	S_2	S_1

State Diagram



https://en.wikipedia.org/wiki/State_transition_table

Mathematical Models for acceptors

A **deterministic finite state machine** or **acceptor** deterministic finite state machine is a quintuple (Σ, S, s_0, δ, F), where:

- Σ is the input alphabet (a finite, non-empty set of symbols).
- S is a finite, non-empty set of states.
- s_0 is an initial state, an element of S.
- δ is the state-transition function: $\delta : S \times \Sigma \rightarrow S$
- F is the set of final states, a (possibly empty) subset of S.

https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

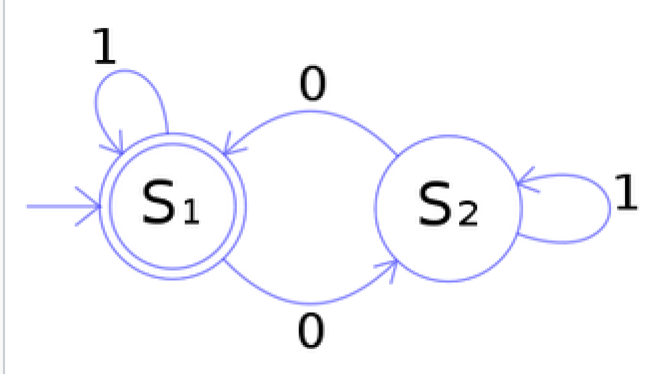
Deterministic Finite Automaton Example (1)

The following example is of a DFA M, with a binary alphabet, which requires that the input contains an even number of 0s.

M = (Q, Σ, δ, q0, F) where

- Q = {S1, S2},
- Σ = {0, 1},
- q0 = S1,
- F = {S1}, and
- δ is defined by the following state transition table:

	0	**1**
$\mathbf{S_1}$	S_2	S_1
$\mathbf{S_2}$	S_1	S_2



The state diagram for *M*

https://en.wikipedia.org/wiki/Deterministic_finite_automaton

Deterministic Finite Automaton Example (2)

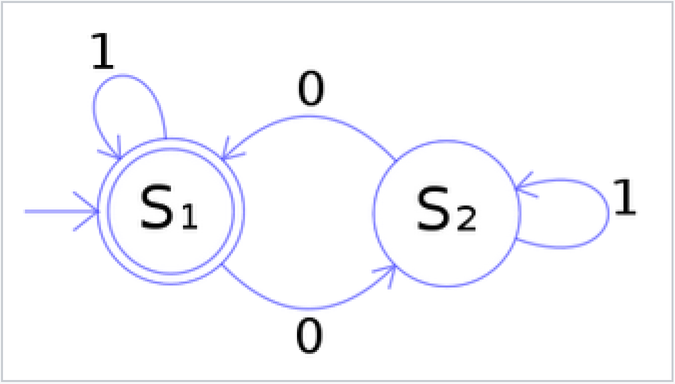
The **state S1** represents that there has been an even number of 0s in the input so far, while **S2** signifies an odd number.

A **1** in the input does not change the state of the automaton.

When the input ends, the state will show whether the input contained an even number of **0**s or not.

If the input did contain an even number of **0**s, M will finish in **state S1**, an accepting state, so the input string will be accepted.

The language recognized by M is the regular language given by the regular expression **((1*) 0 (1*) 0 (1*))***, where "*" is the Kleene star, e.g., **1*** denotes any number (possibly zero) of consecutive **ones**.



The state diagram for *M*

https://en.wikipedia.org/wiki/Deterministic_finite_automaton

Mathematical Model for transducers (1)

A **finite-state transducer** is a sextuple (Σ, Γ, S, s0, δ, ω), where:
Σ is the input alphabet (a finite non-empty set of symbols).
Γ is the output alphabet (a finite, non-empty set of symbols).
S is a finite, non-empty set of states.
s0 is the initial state, an element of S.
ω is the output function.

https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Mathematical Model for transducers (2)

If the **output** function is a function of a **state** and **input** alphabet ($\omega : S \times \Sigma \rightarrow \Gamma$) that definition corresponds to the **Mealy model**, and can be modelled as a Mealy **machine**.

If the **output** function depends only on a **state** ($\omega : S \rightarrow \Gamma$) that definition corresponds to the **Moore model**, and can be modelled as a **Moore machine**.

A finite-state machine with no output function at all is known as a **semiautomaton** or **transition** system.

https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

References

[1] http://en.wikipedia.org/
[2]