Finite State Machine (1A)

Young Won Lim
6/9/18

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

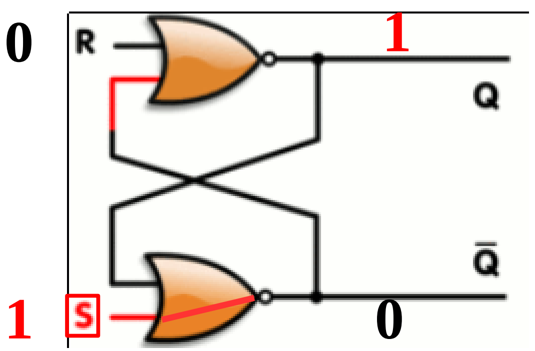
This document was produced by using LibreOffice and Octave.

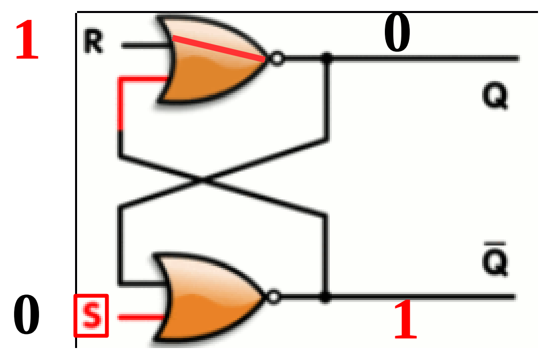
FSM and Digital Logic Circuits

- 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

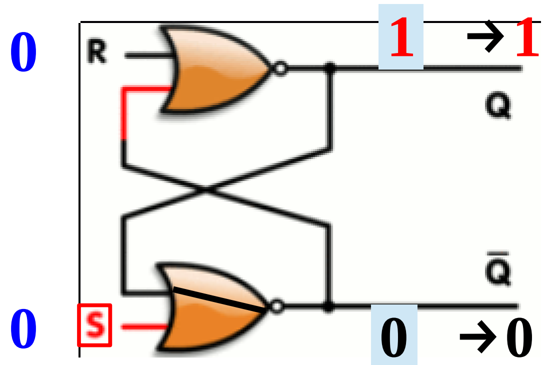
Q=1
$\overline{Q}$=0

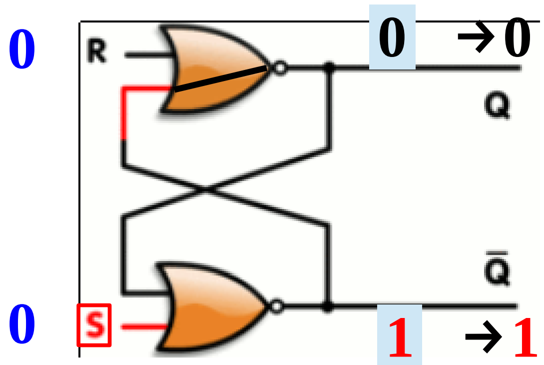
RESET S=0 R=**1**

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 $\overline{Q}$

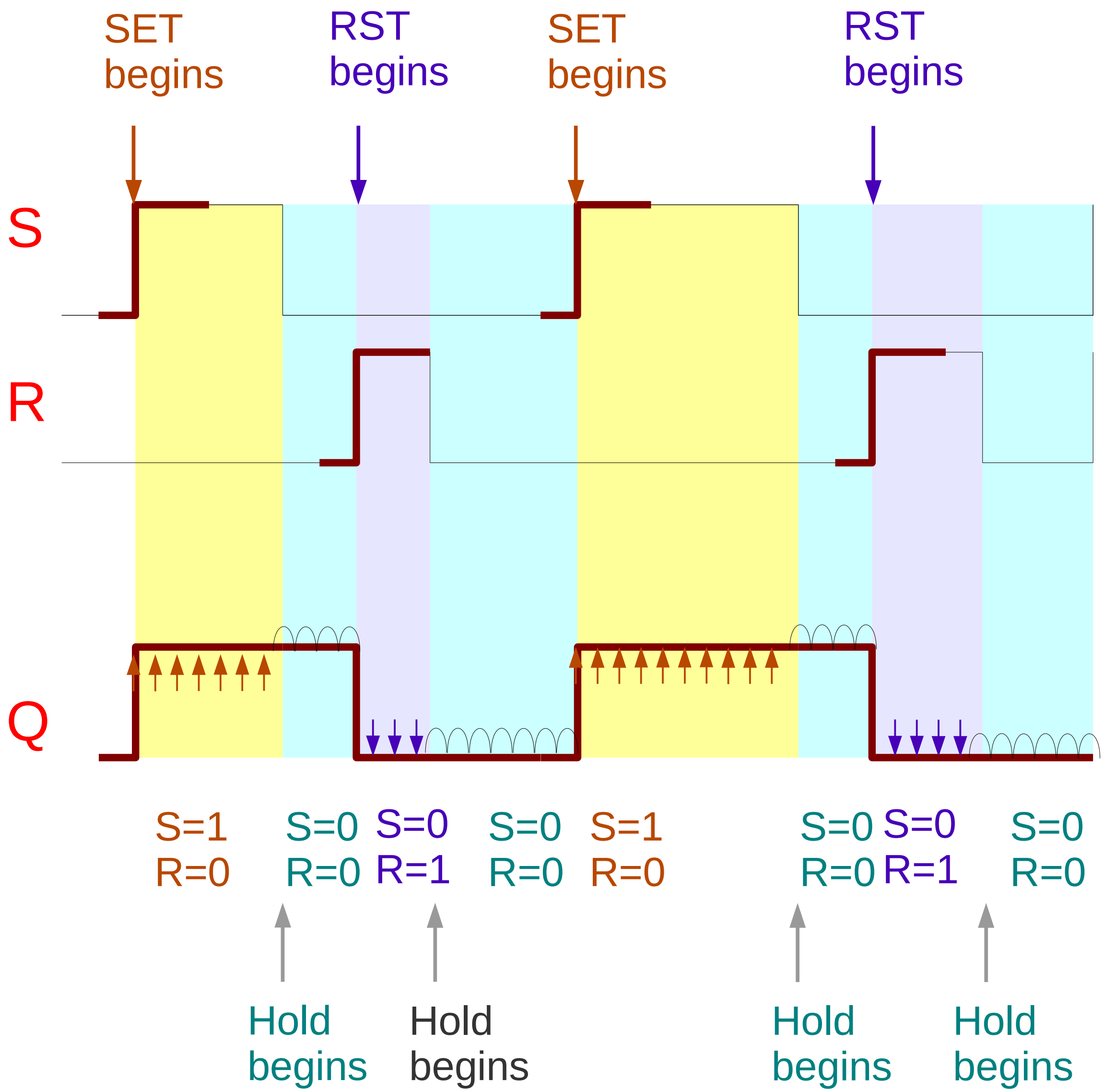
HOLD	S=0 R=0

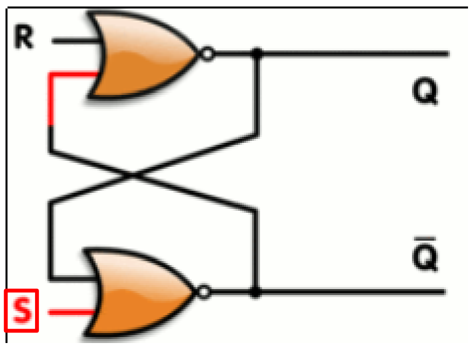
Q=old Q

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

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

NOR-based SR Latch





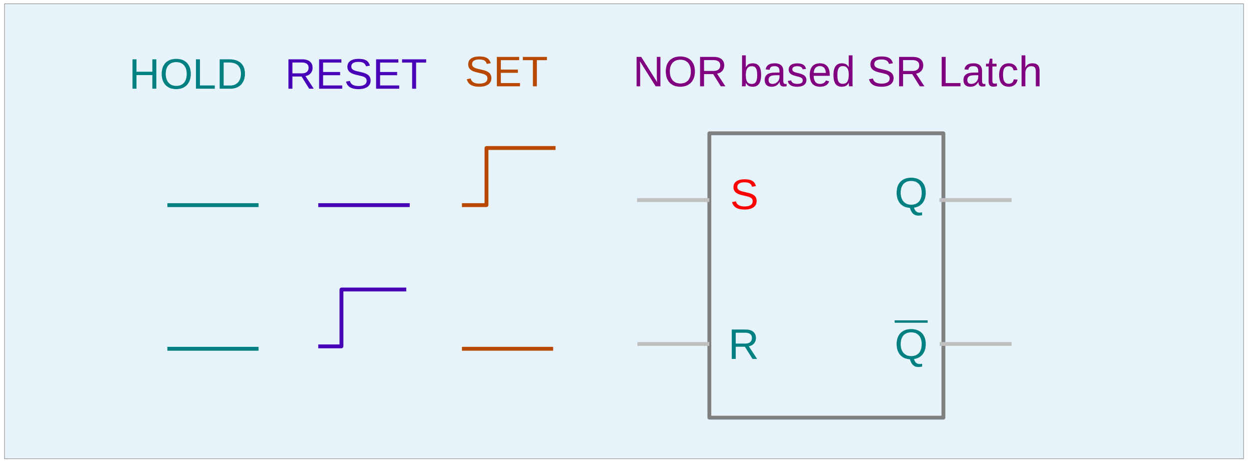
SET: S=**1**, R=0 → Q=1, $\overline{Q}$=0

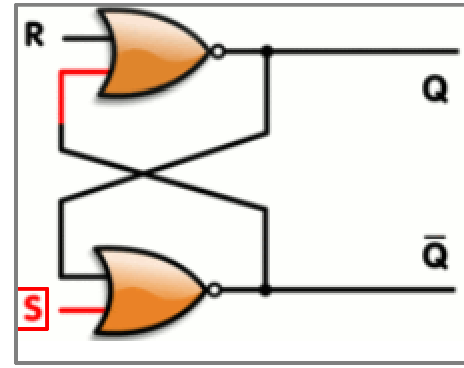
RESET: S=0, R=**1** → Q=0, $\overline{Q}$=1

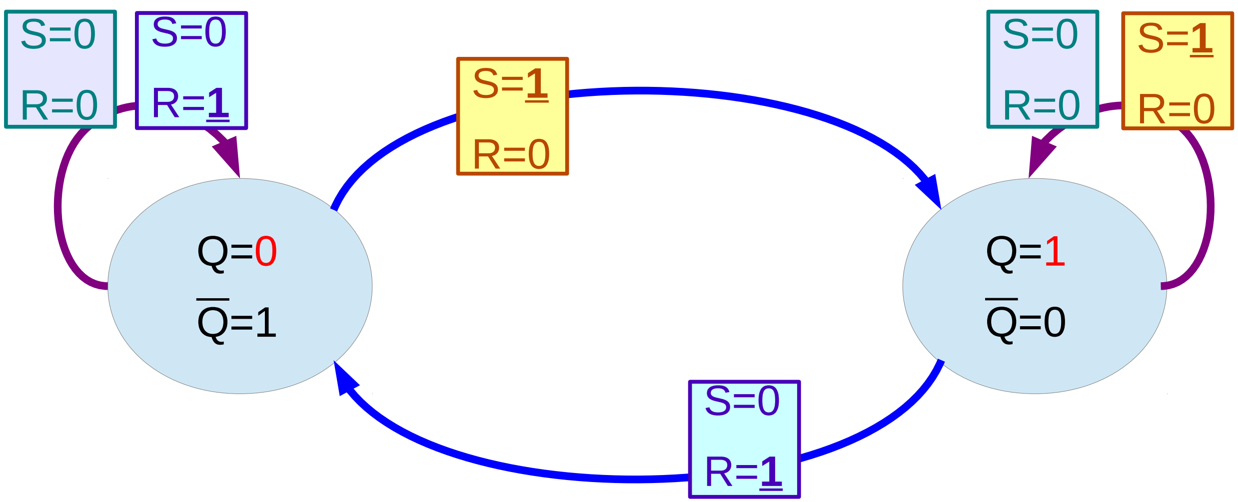
HOLD: S=0, R=0 → Q=old Q, $\overline{Q}$=old $\overline{Q}$

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

NOR-based SR Latch States



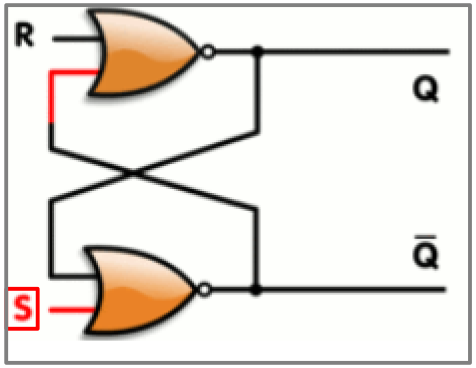


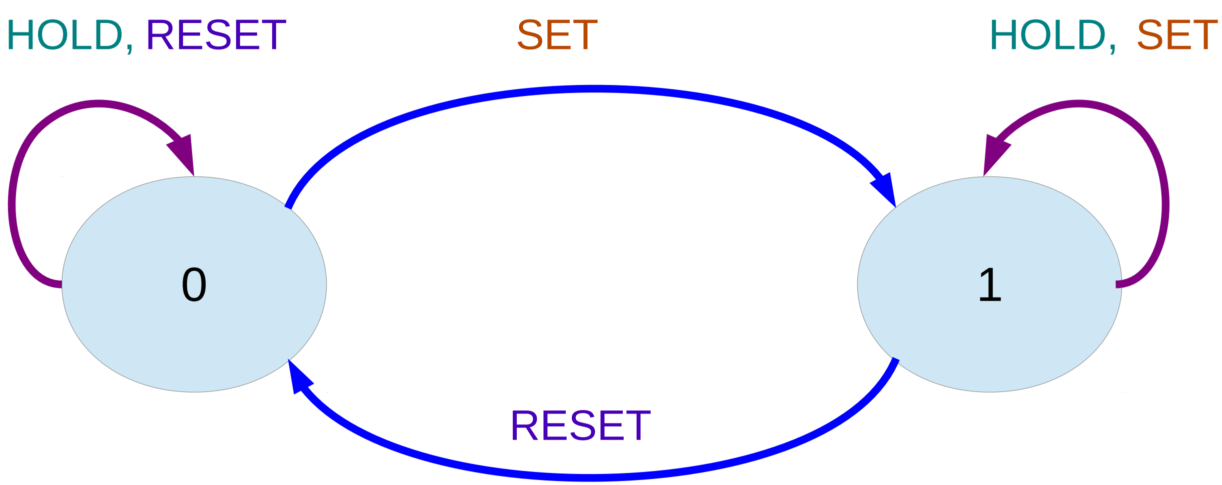


SET	S=1 R=0	Q=1 $\overline{Q}$=0
RESET	S=0 R=1	Q=0 $\overline{Q}$=1
HOLD	S=0 R=0	Q=old Q $\overline{Q}$=old $\overline{Q}$

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

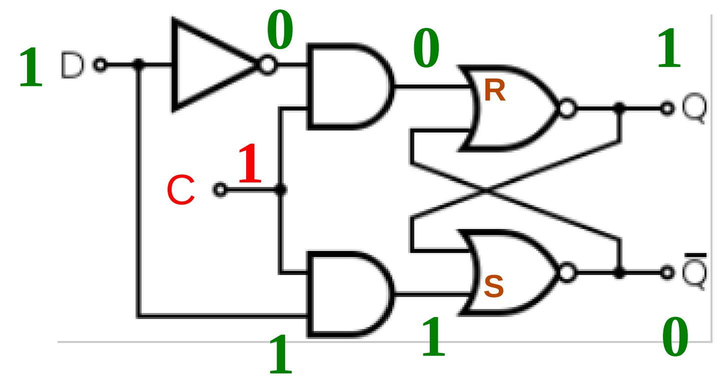
HOLD S=0 R=0 — Q=old Q, $\overline{Q}$=old $\overline{Q}$

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

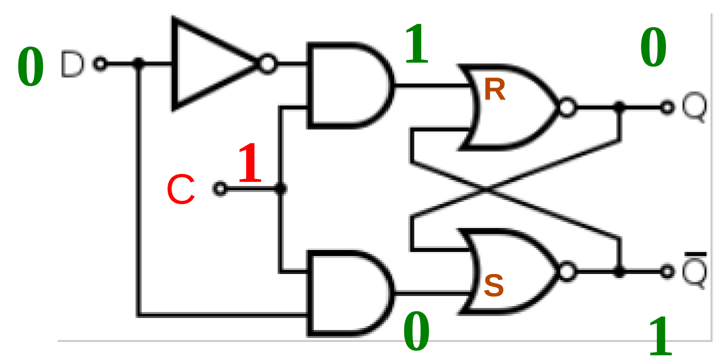
NOR-based D Latch – SET / RESET

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

D=**1**	SET	S=**1**	Q=1
C=1		R=0	$\overline{Q}$=0

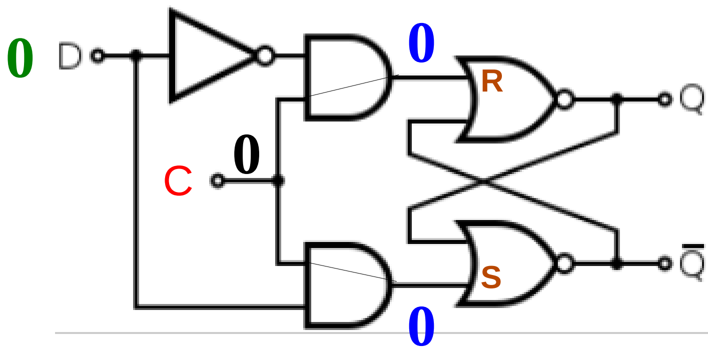


D=**0**	RESET	S=0	Q=0
C=1		R=**1**	$\overline{Q}$=1



NOR-based D Latch – HOLD

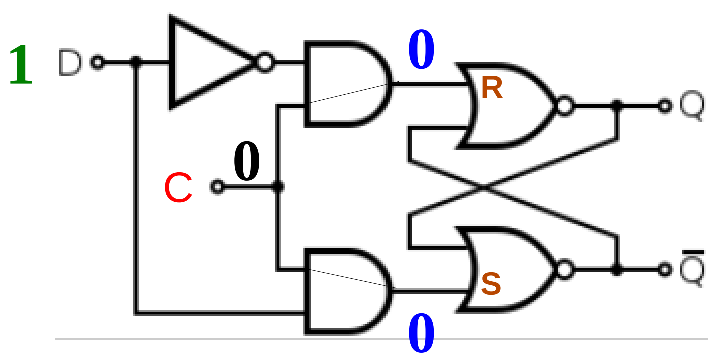
https://en.wikipedia.org/wiki/Flip-flop_(electronics)



D=**X**
C=0

HOLD S=0 R=0

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



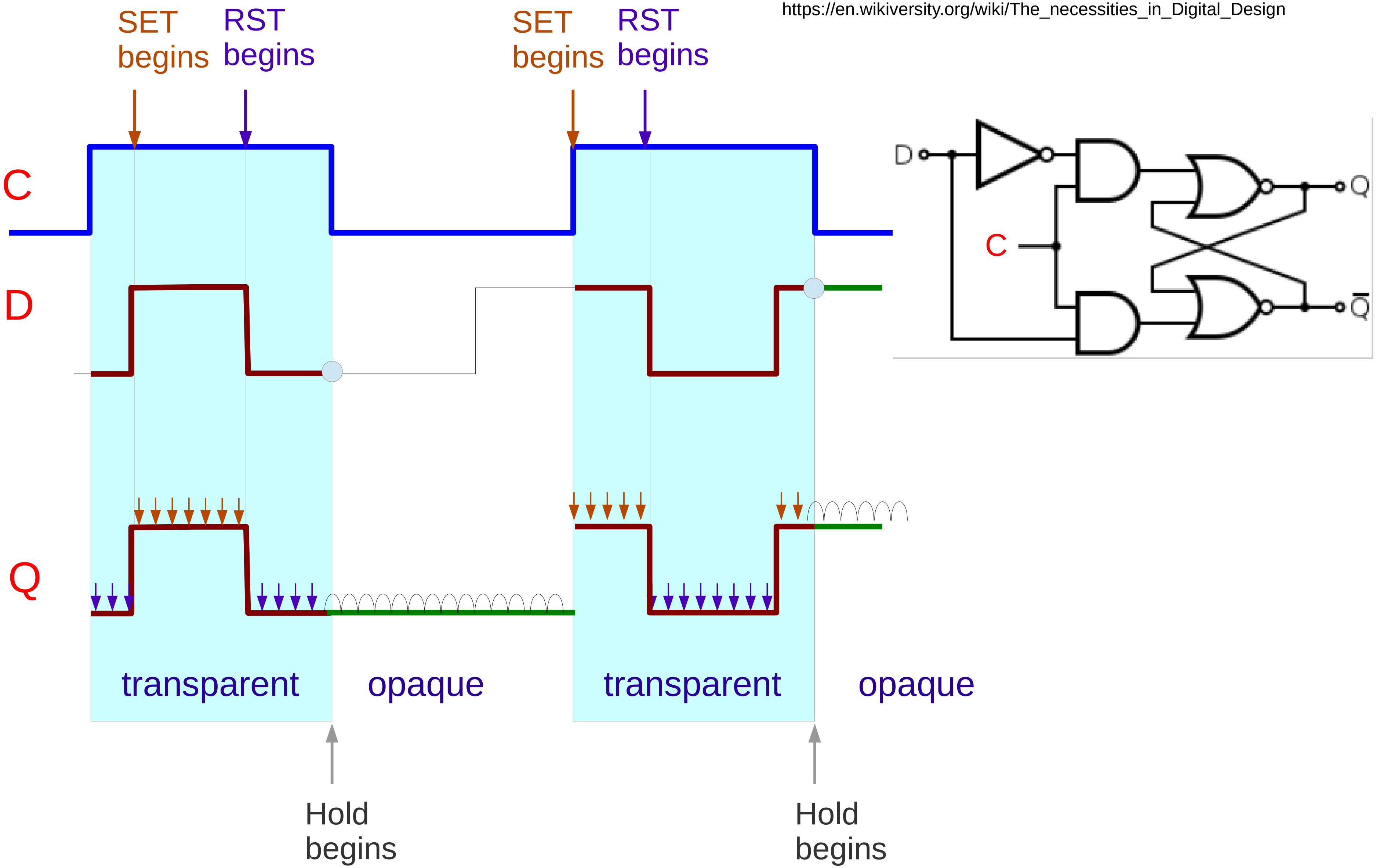
D=**X**
C=0

HOLD S=0 R=0

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

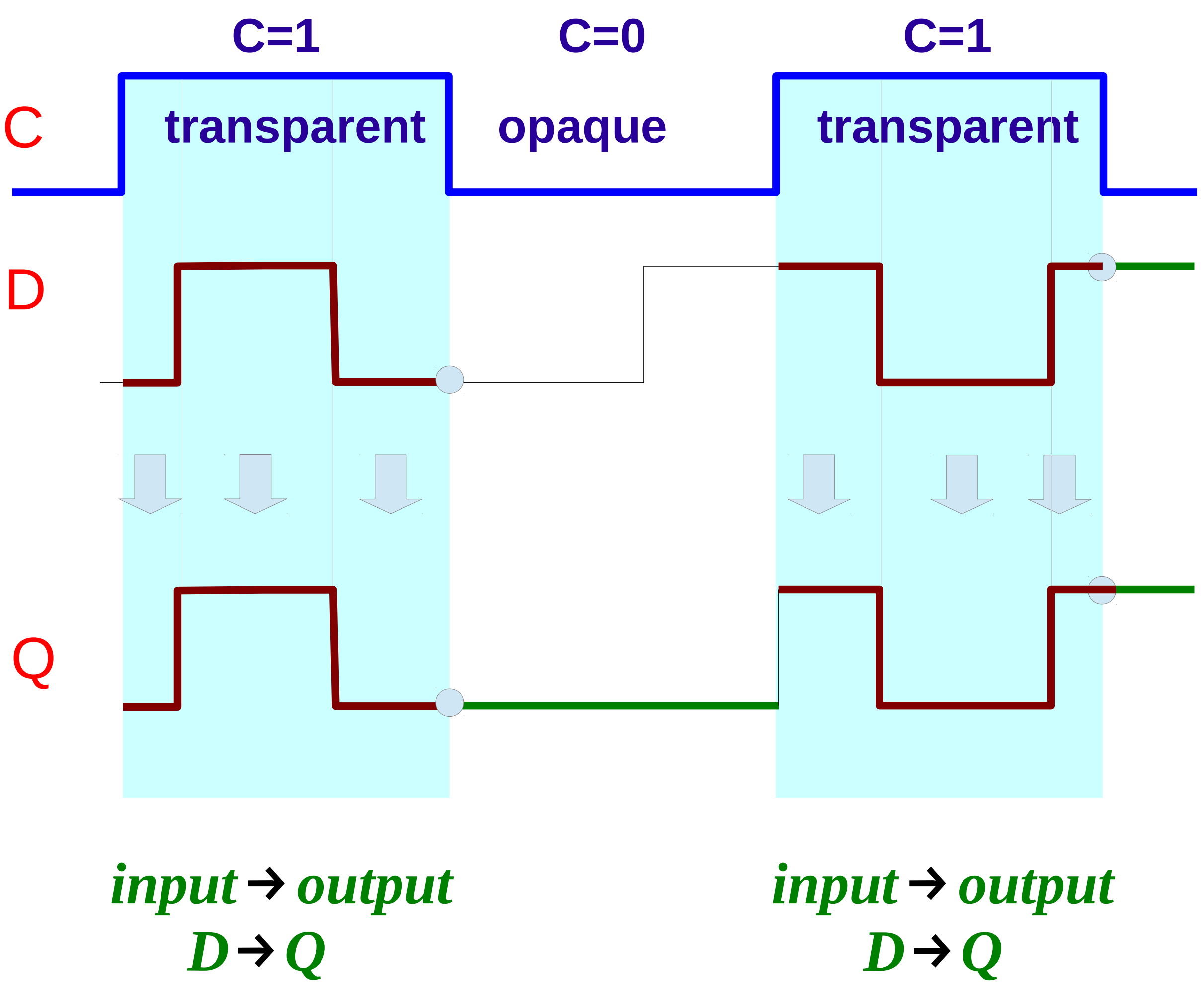
NOR-based D Latch – Set / Reset / Hold

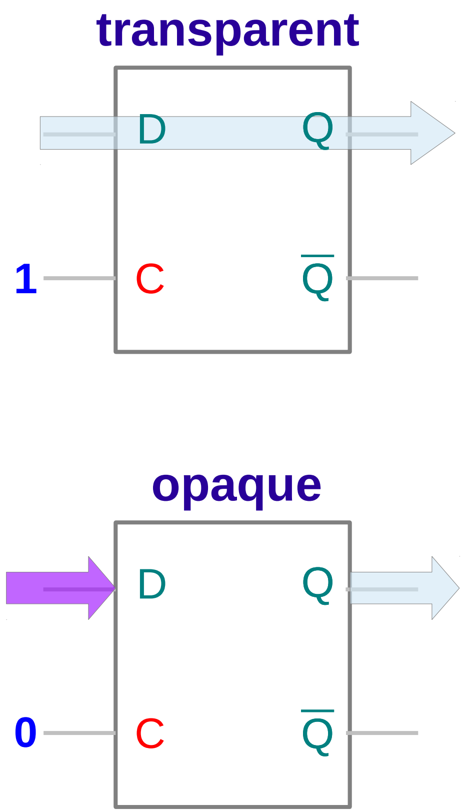
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design



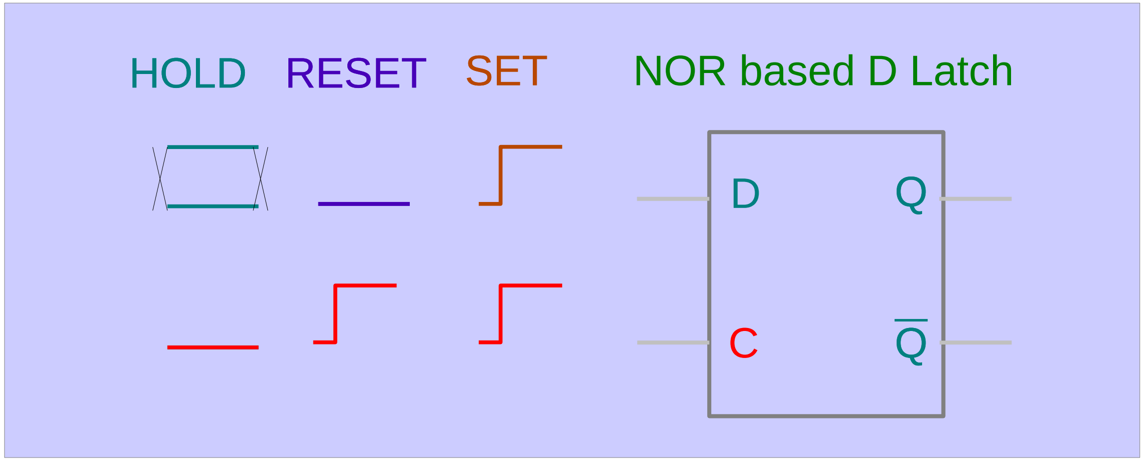
NOR-based D Latch – transparent / opaque

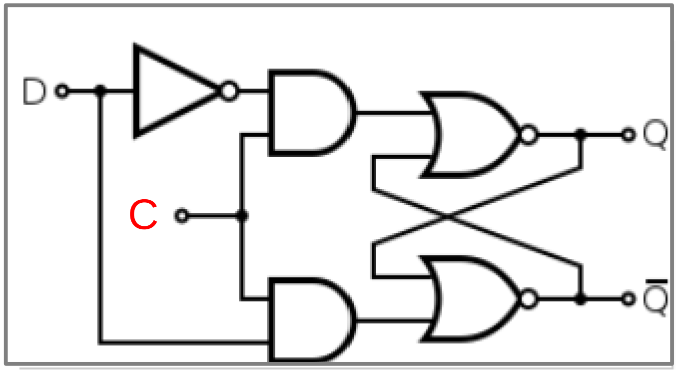
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

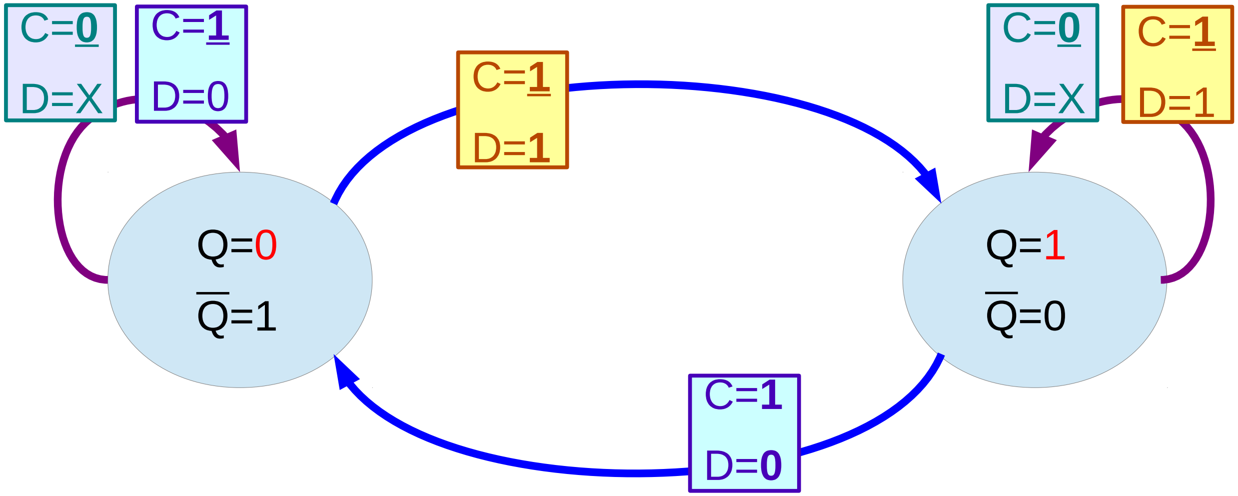




NOR-based D Latch States

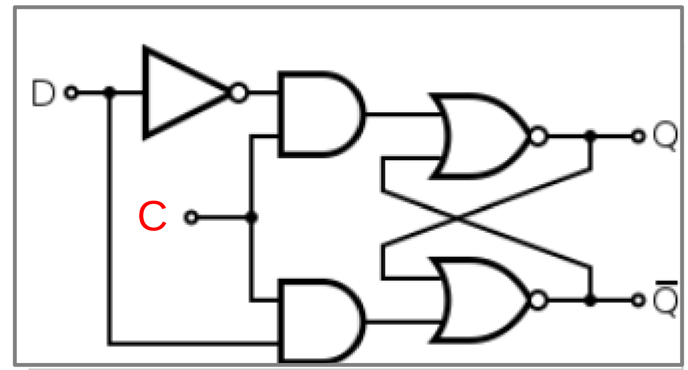


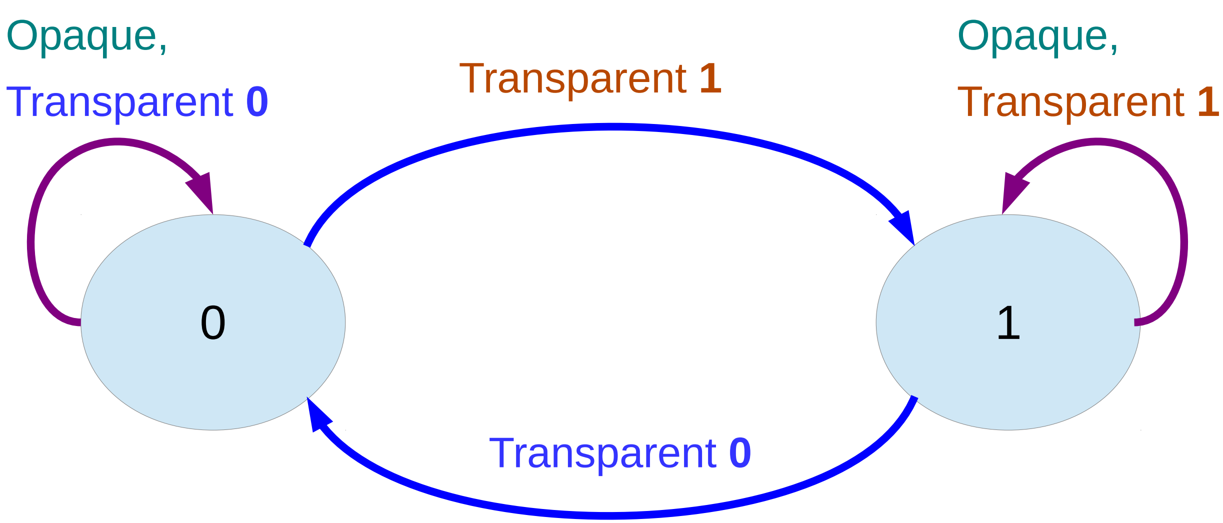




https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

D Latch States

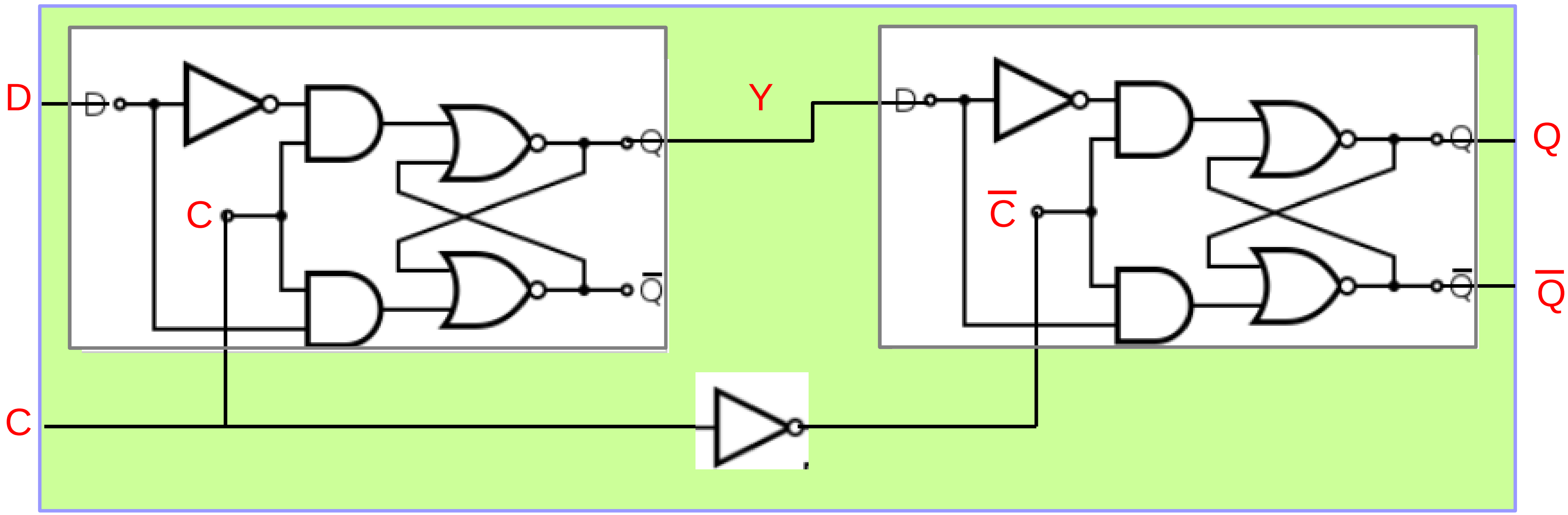


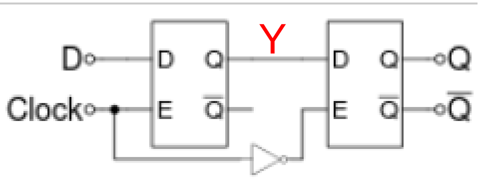


Trans 1	C=**1**, D=**1**	Q=1, $\overline{Q}$=0
Trans 0	C=**1**, D=0	Q=0, $\overline{Q}$=1
Opaque	C=**0**, D=X	Q=old Q, $\overline{Q}$=old $\overline{Q}$

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

Master-Slave FlipFlops

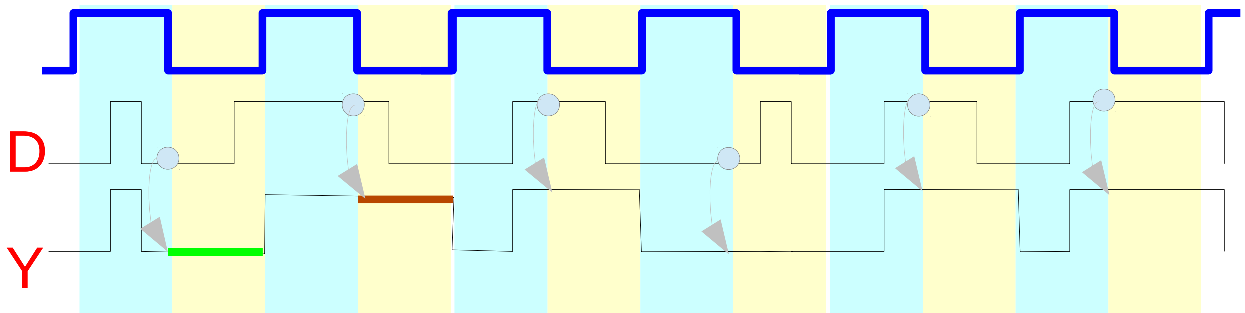


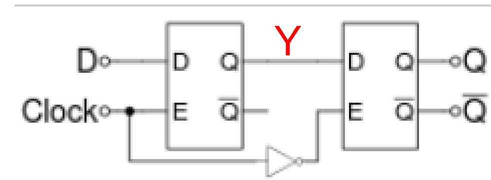


https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

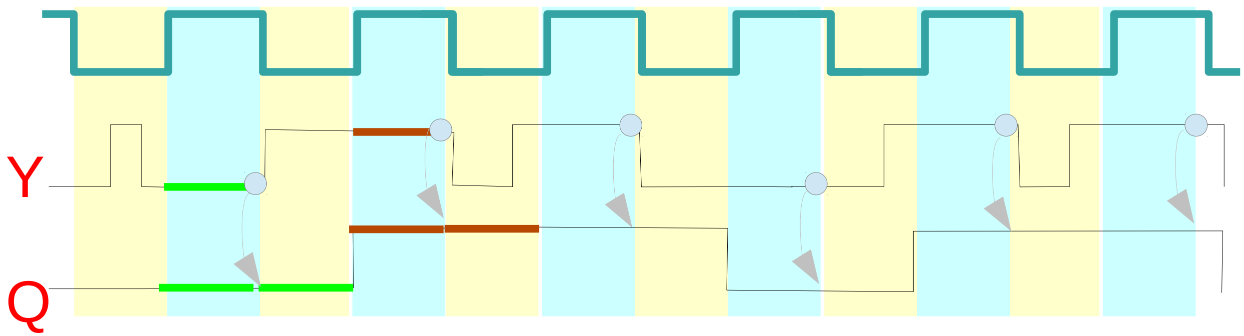
Master-Slave D FlipFlop

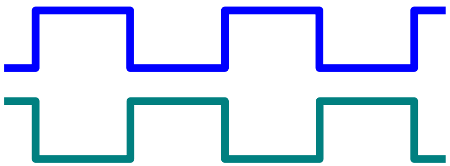
Master D Latch

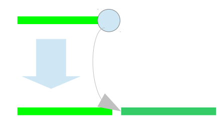




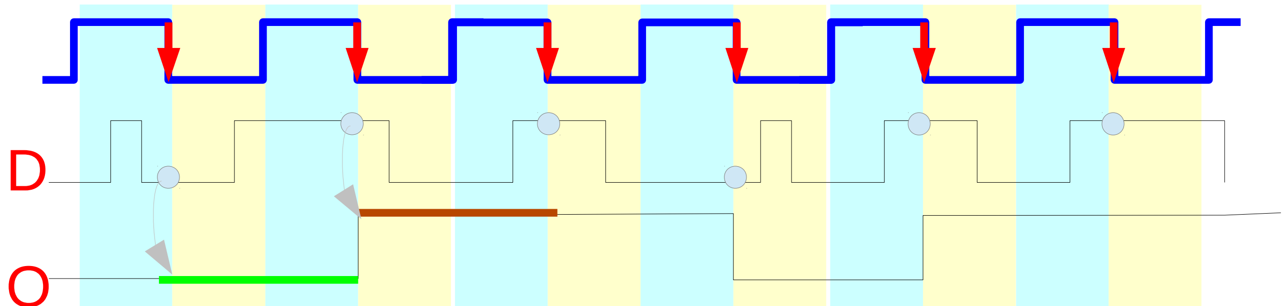
Slave D Latch







Master-Slave D F/F

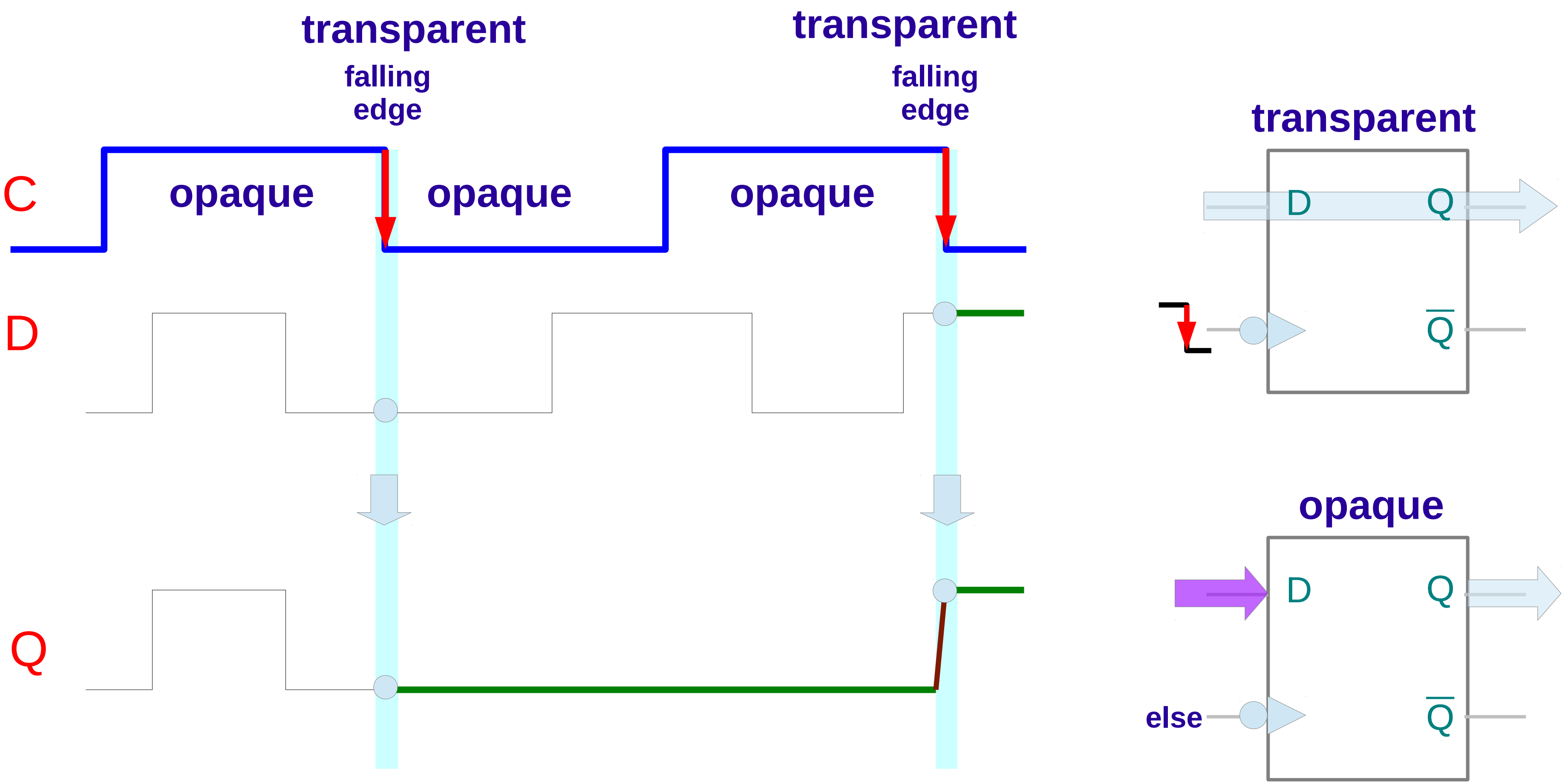


the hold output of the master is transparently reaches the output of the slave

this value is held for another half period

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

Master Slave D FlipFlop – transparent / opaque



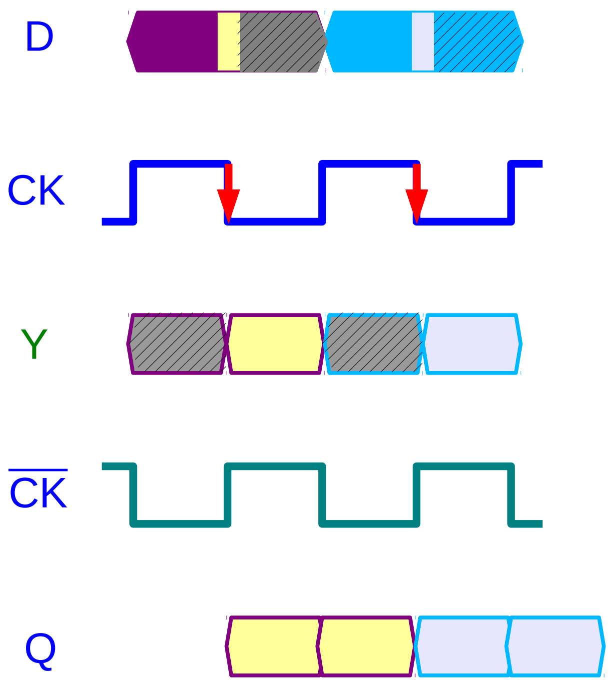
input → output
D → Q

input → output
D → Q

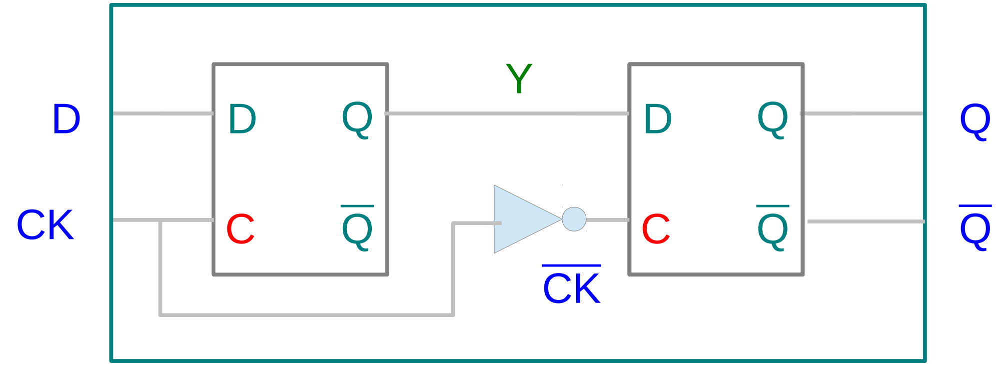
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

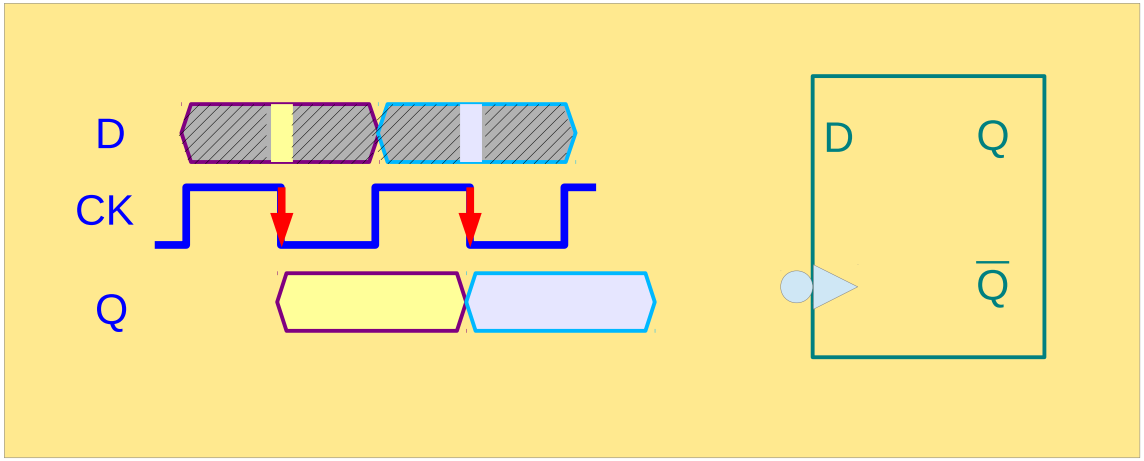
Master-Slave D FlipFlop – Falling Edge

Master D Latch



Slave D Latch



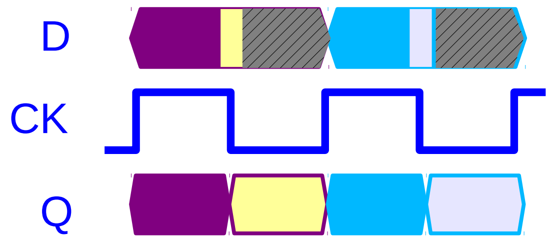


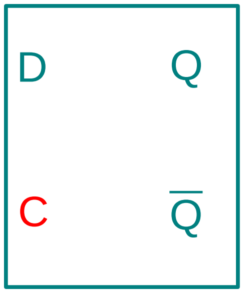
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

D Latch & D FlipFlop

Level Sensitive D Latch

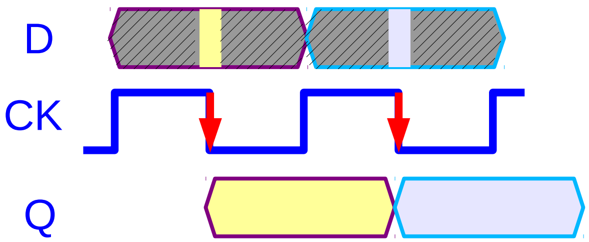
CK=1 transparent
CK=0 opaque

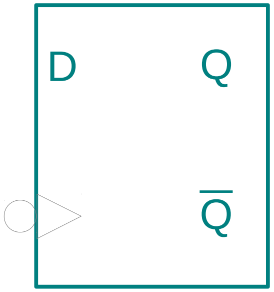




Edge Sensitive D FlipFlop

CK=1→0 transparent
else opaque

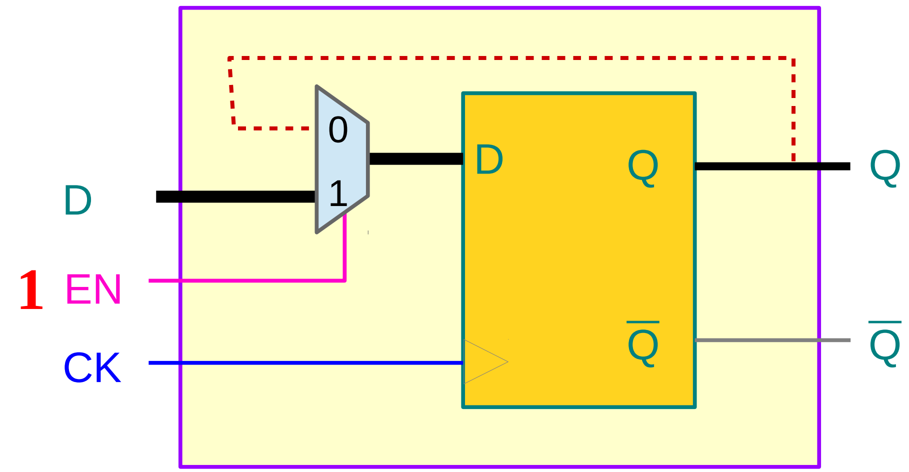




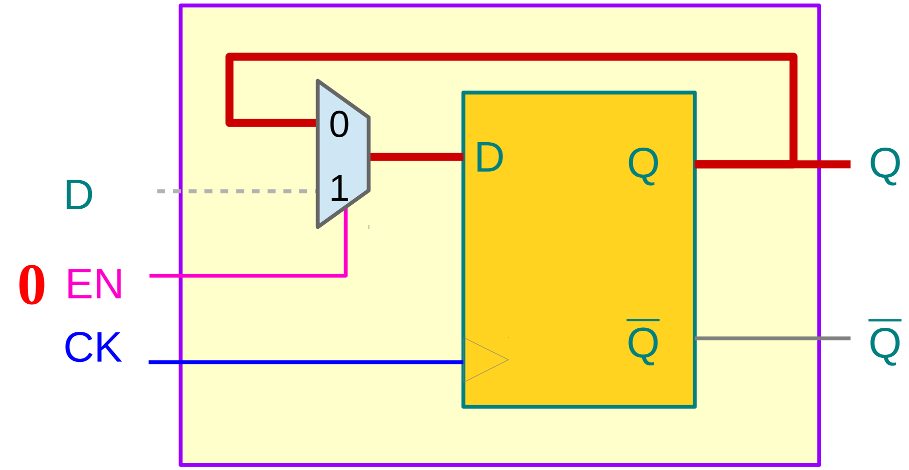
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

D FlipFlop with Enable (1)

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

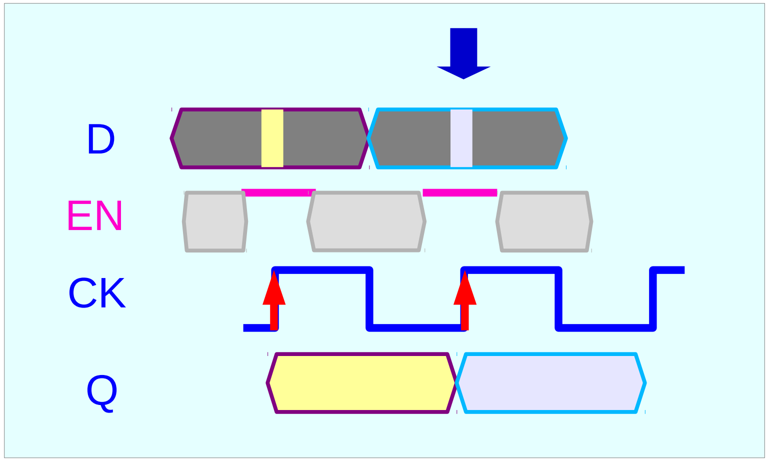


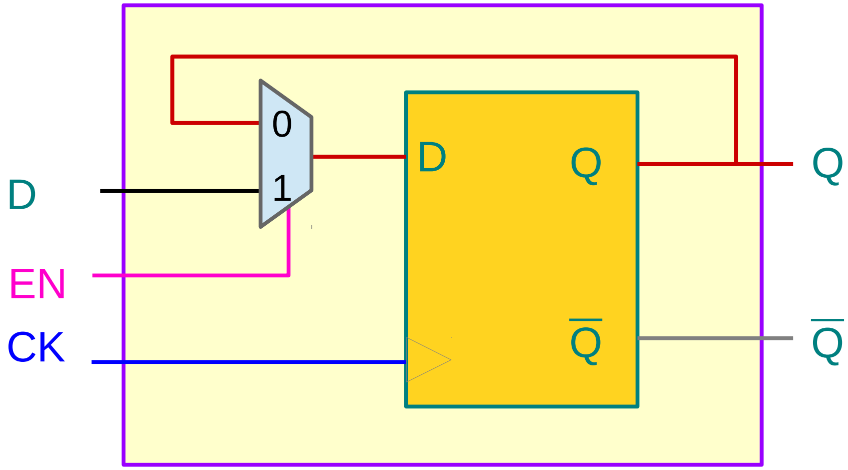
EN=0 Holding D Flip Flop
Sampling **Q** output
@ **posedge** of CK

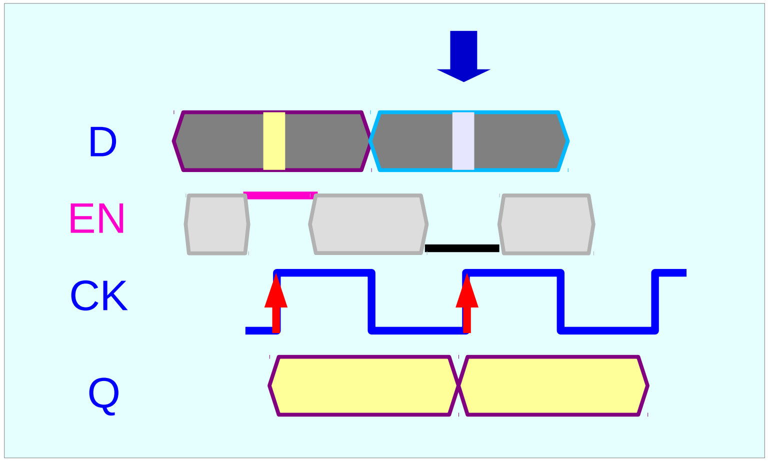


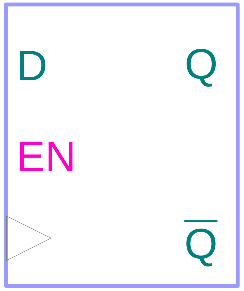
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

D FlipFlop with Enable (2)



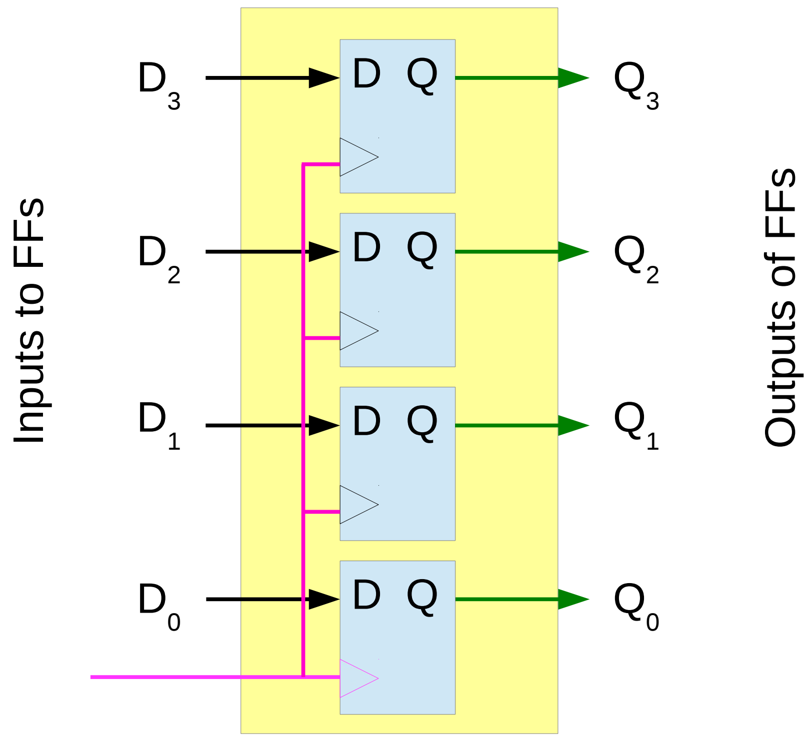


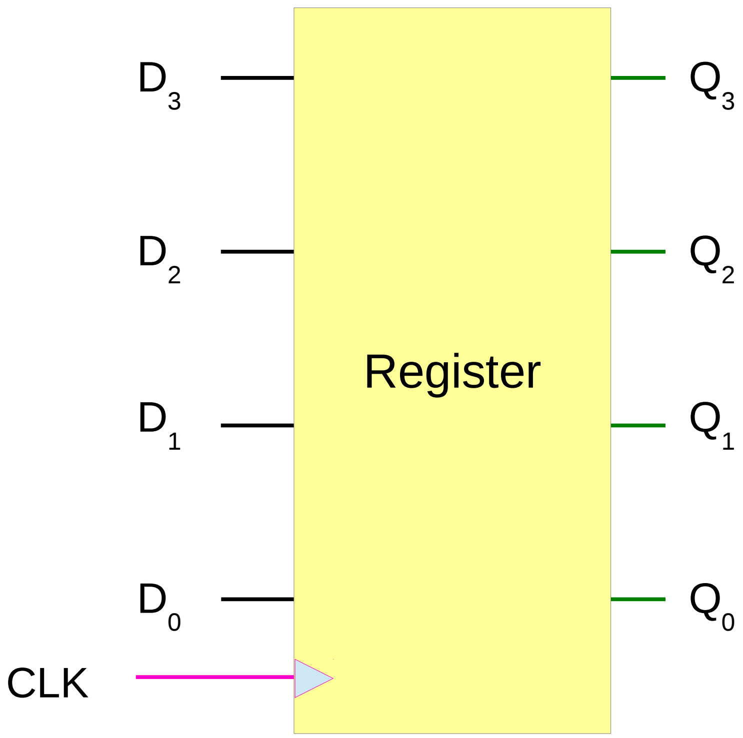




https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

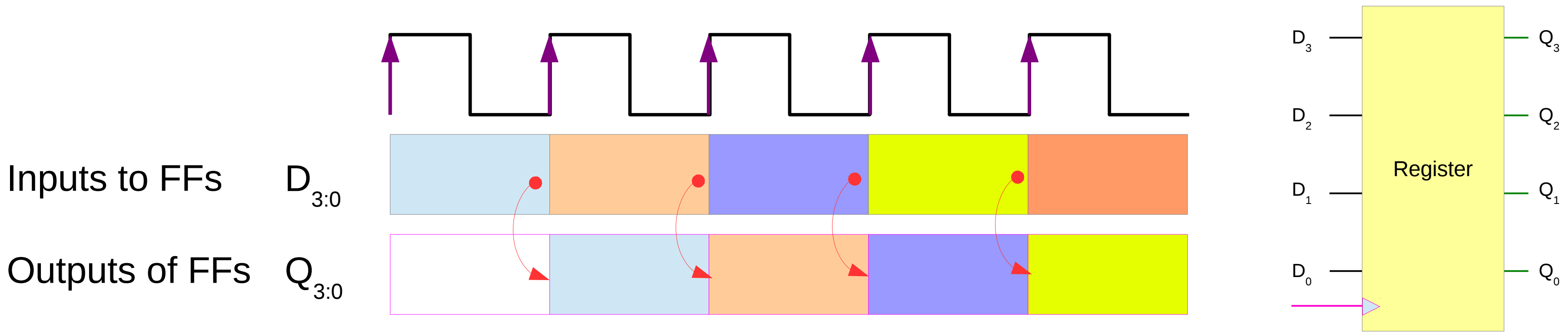
Registers





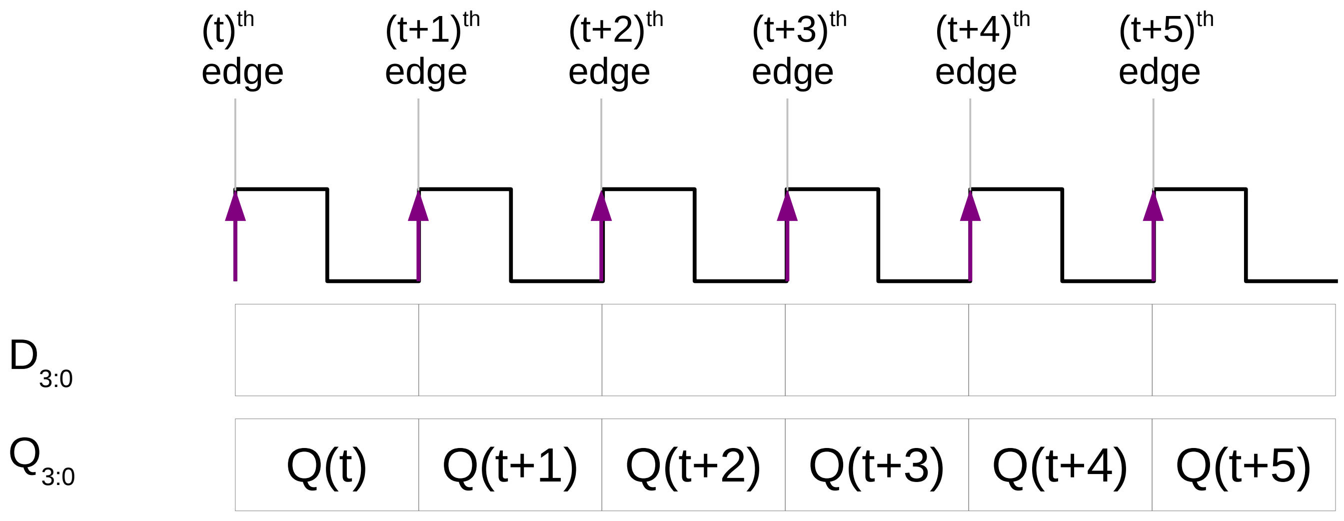
https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

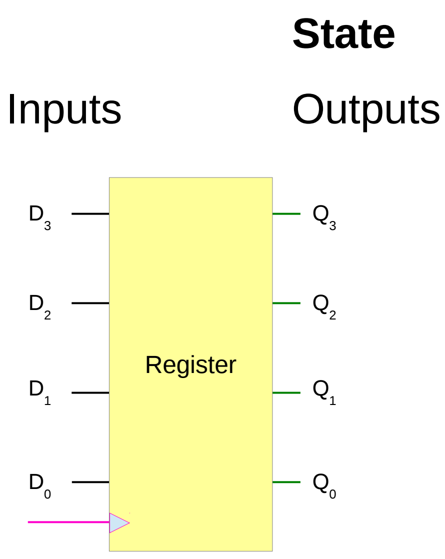
FF Timing (Ideal)



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

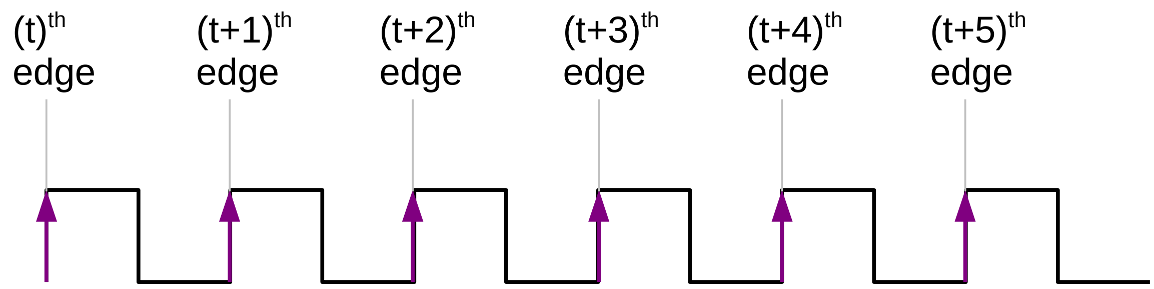
States



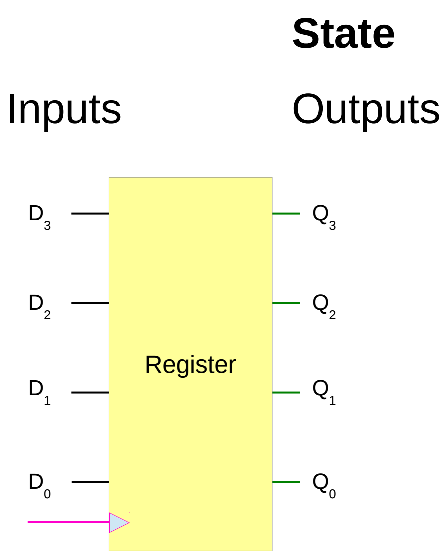


https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Sequence of States



$D_{3:0}$	?	?	?	?	?	?
$Q_{3:0}$	Q(t)	Q(t+1)	Q(t+2)	Q(t+3)	Q(t+4)	Q(t+5)

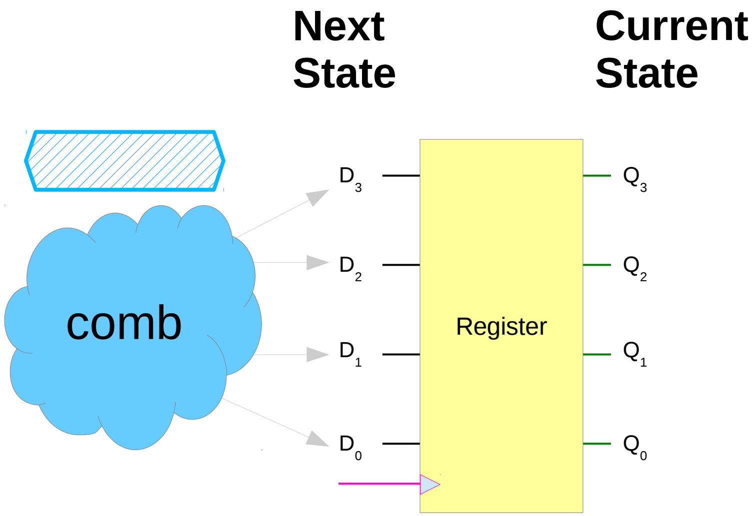


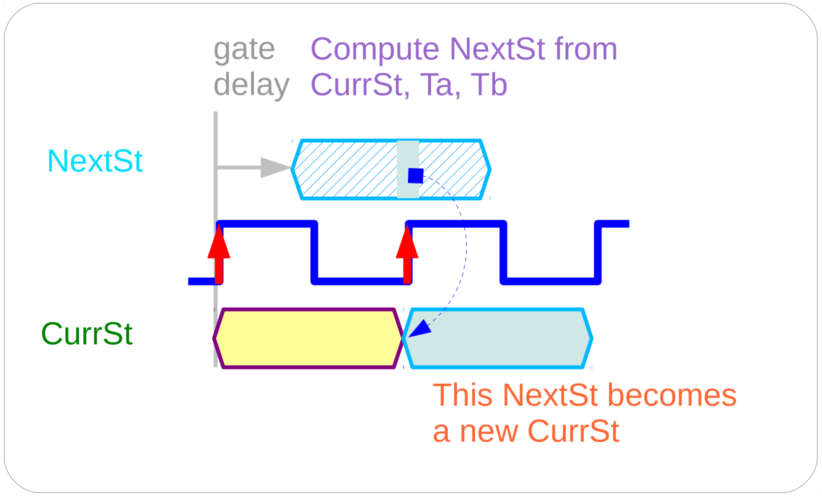
Find inputs to FFs

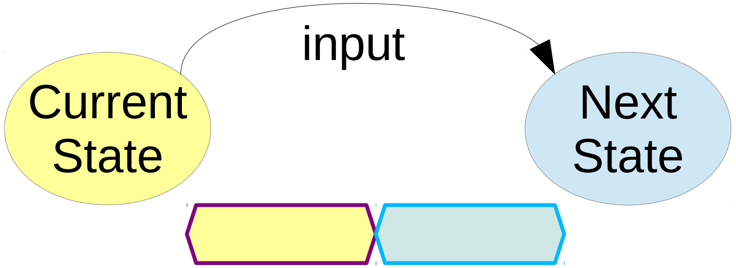
which will make outputs
in this sequence

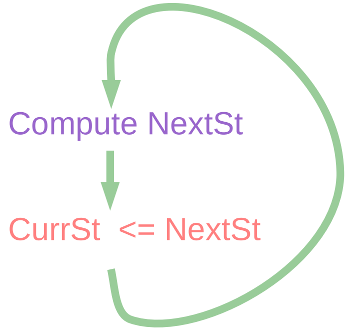
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

How to change current state



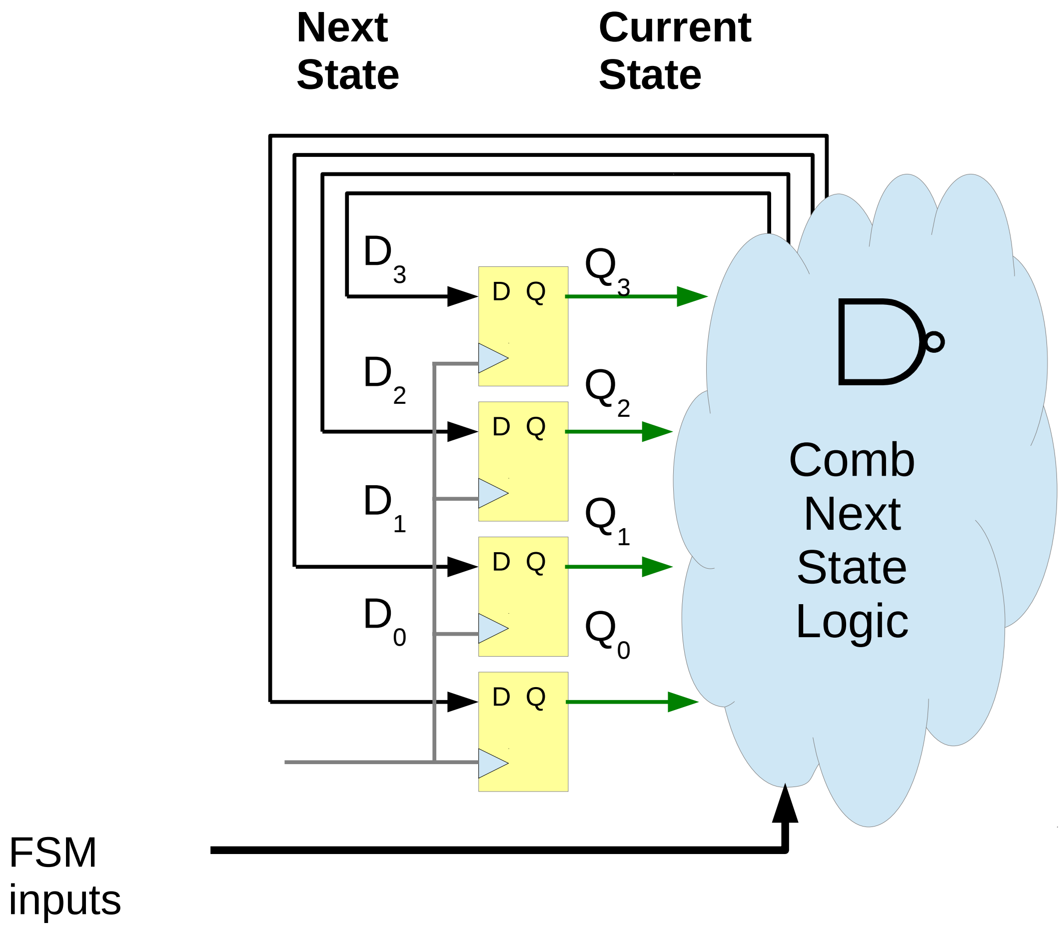






https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Finding FF Inputs



During the t^{th} clock edge period,

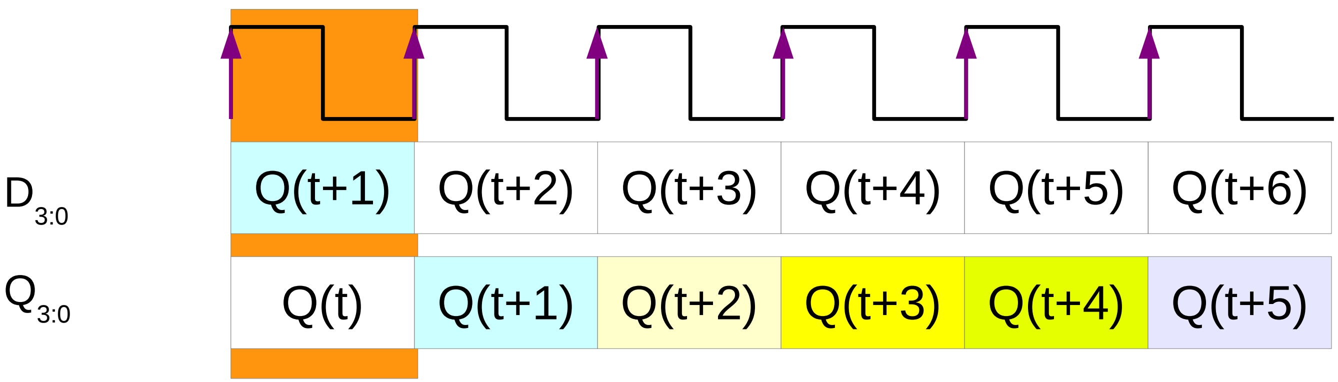
Compute the next state Q(t+1) using the current state Q(t) and other external inputs

Place it to FF inputs

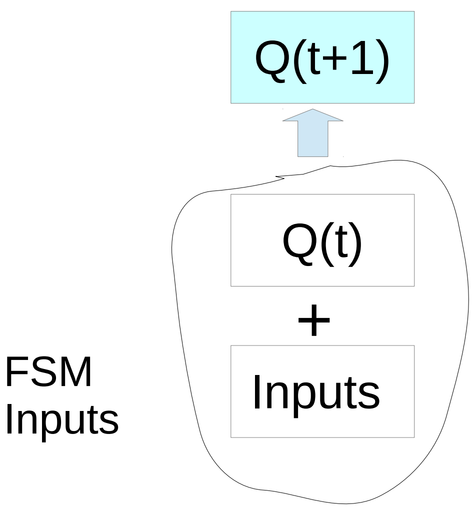
After the next clock edge, $(t+1)^{th}$, the computed next state Q(t+1) becomes the current state

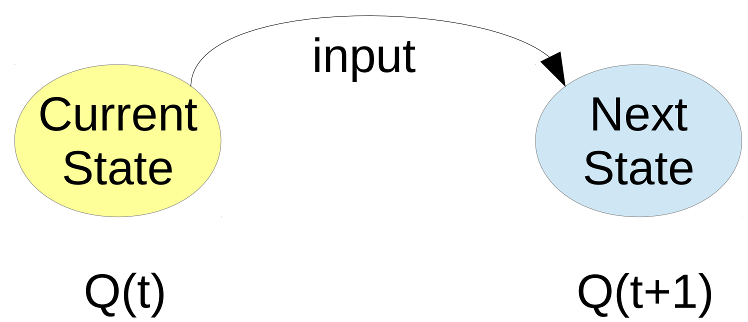
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Method of Finding FF Inputs

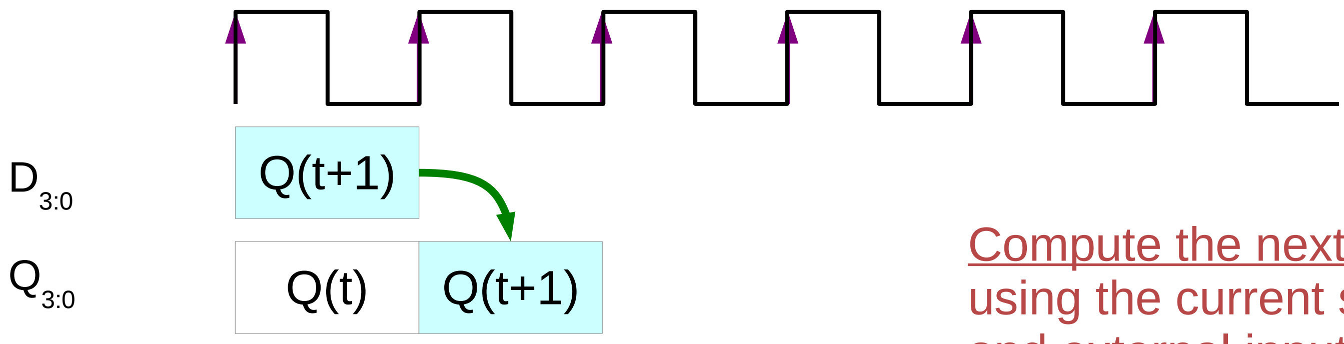


Find the boolean functions
D3, D2, D1, D0
in terms of Q3, Q2, Q1, Q0,
and external FSM inputs
for all possible cases.

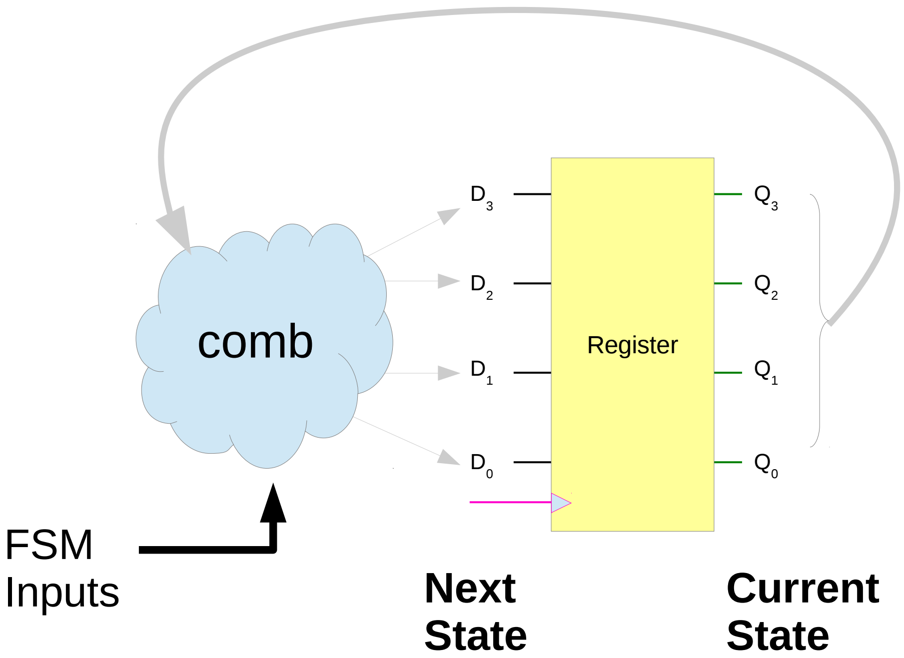


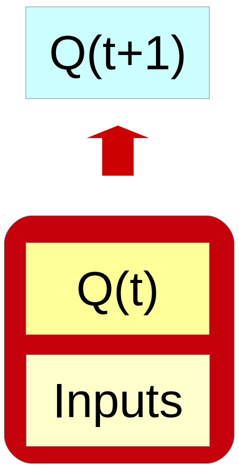


State Transition



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





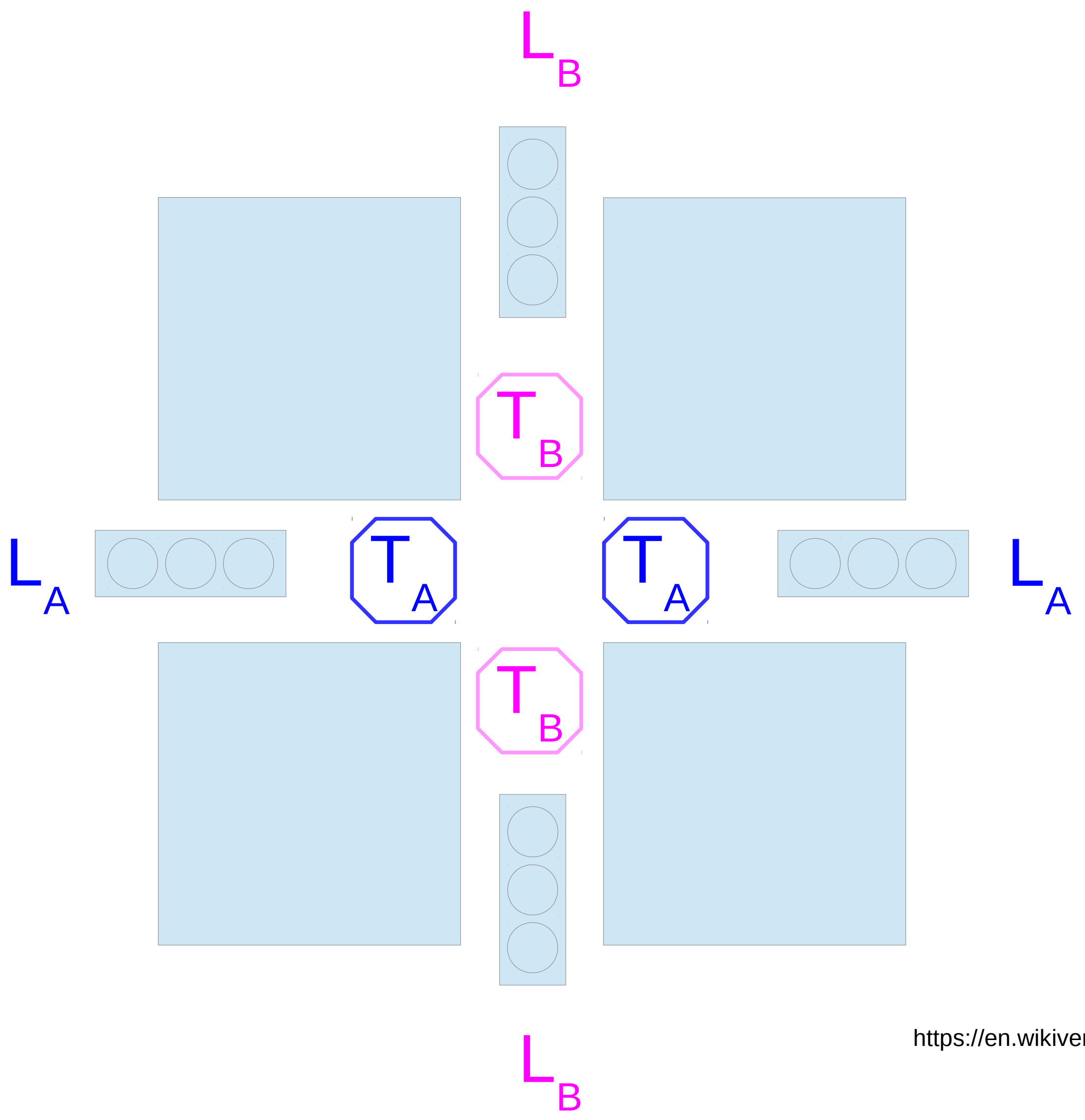
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

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

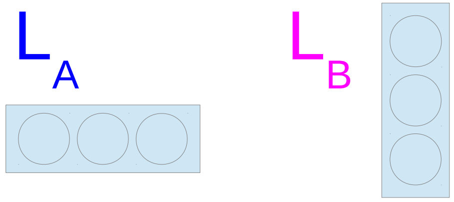
Traffic Lights Example

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

FSM Inputs and Outputs



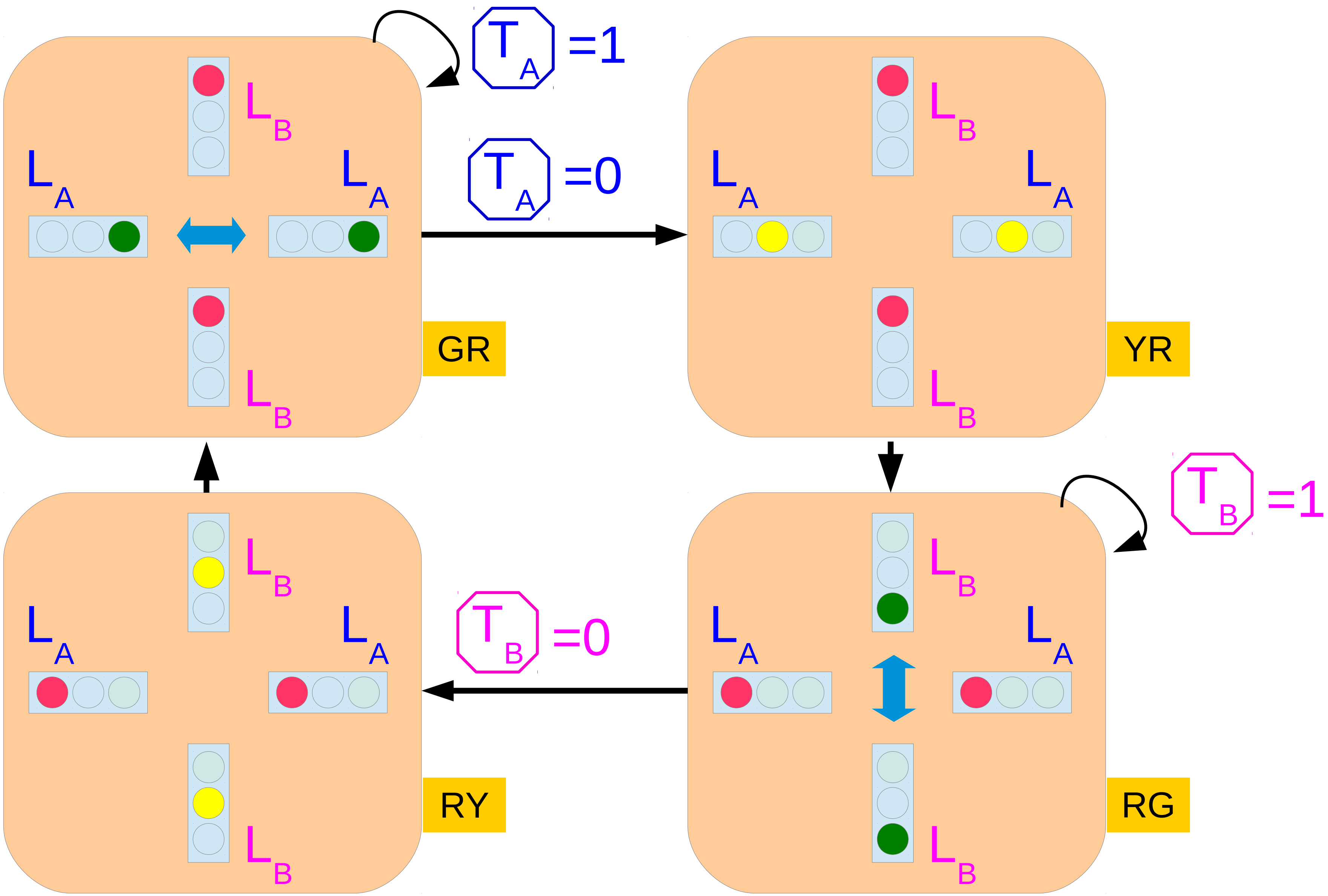
Traffic Lights - **Outputs**



Sensor - **Inputs**

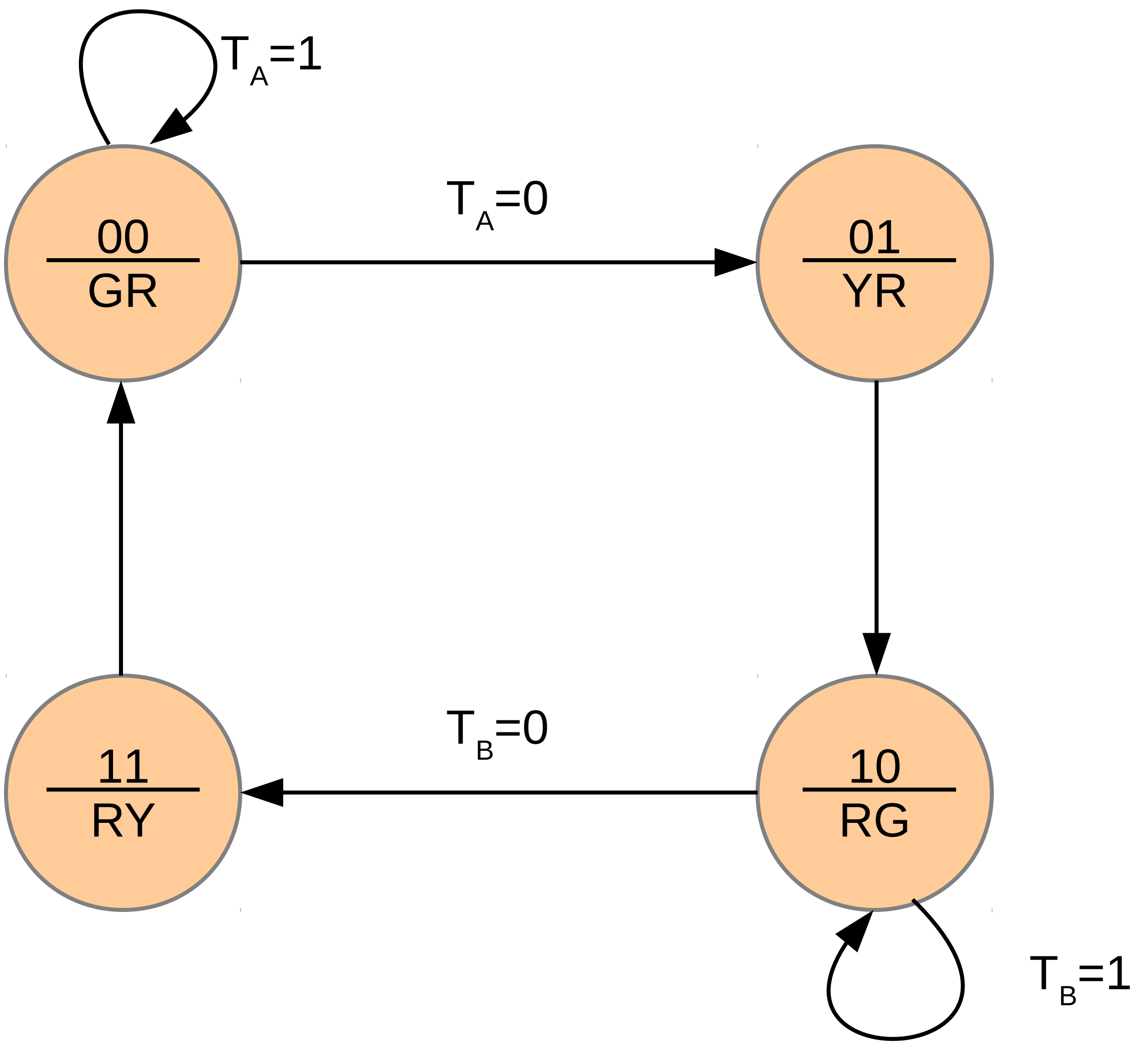
https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

Four States



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

State Transition Diagrams and Tables



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	G	R
0	1	0	1	1	0	Y	R
1	0	1	0	0	0	R	G
1	1	1	0	0	1	R	Y

- G:00
- Y:01
- R:10

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

$$S'_1 = S_1 + S_0$$

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

	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

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

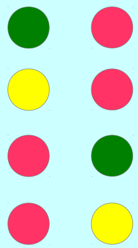
	S_1	S_0	T_A	T_B	S'_0
$\overline{S_1}\,\overline{S_0}\,\overline{T_A}$	0	0	0	X	1
	0	0	1	X	0
	0	1	X	X	0
$S_1\overline{S_0}\,\overline{T_B}$	1	0	X	0	1
	1	0	X	1	0
	1	1	X	X	0

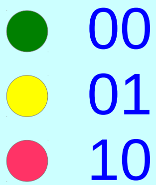
$$S'_0 = \overline{S_1}\,\overline{S_0}\,\overline{T_A} + S_1\overline{S_0}\,\overline{T_B}$$

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





$L_{A1}=S_1$
$L_{A0}=\overline{S_1} S_0$
$L_{B1}=\overline{S_1}$
$L_{B0}=S_1 S_0$

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

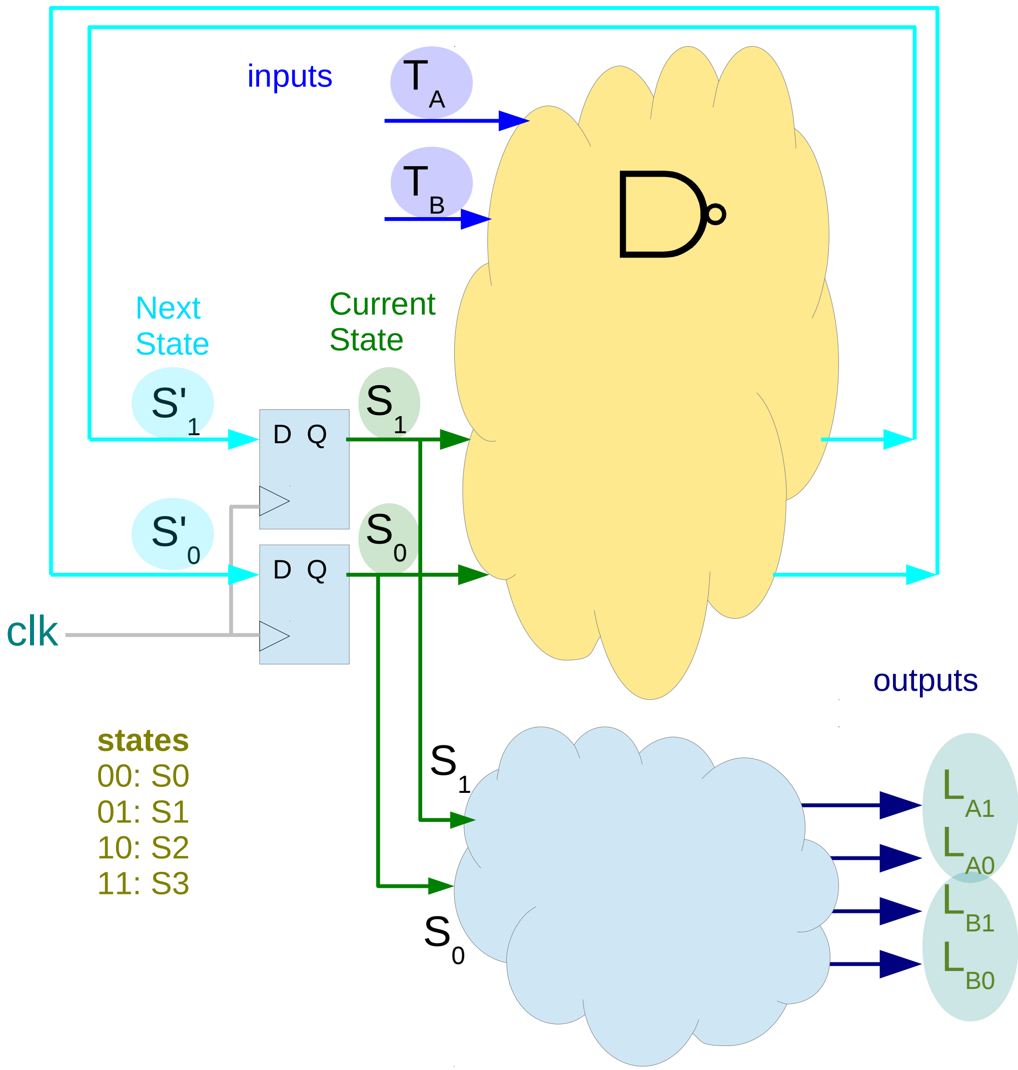
$L_{B1}=\overline{S_1}$

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

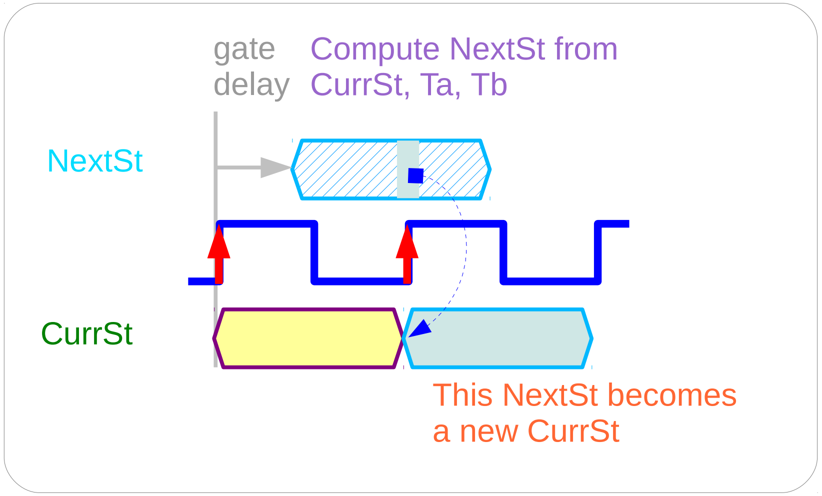
$L_{B0}=S_1 S_0$

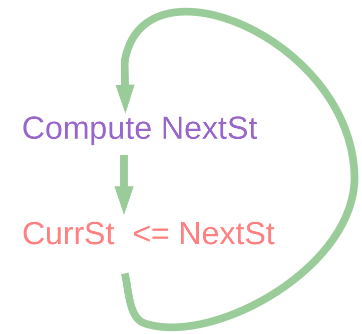
https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

Moore FSM



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



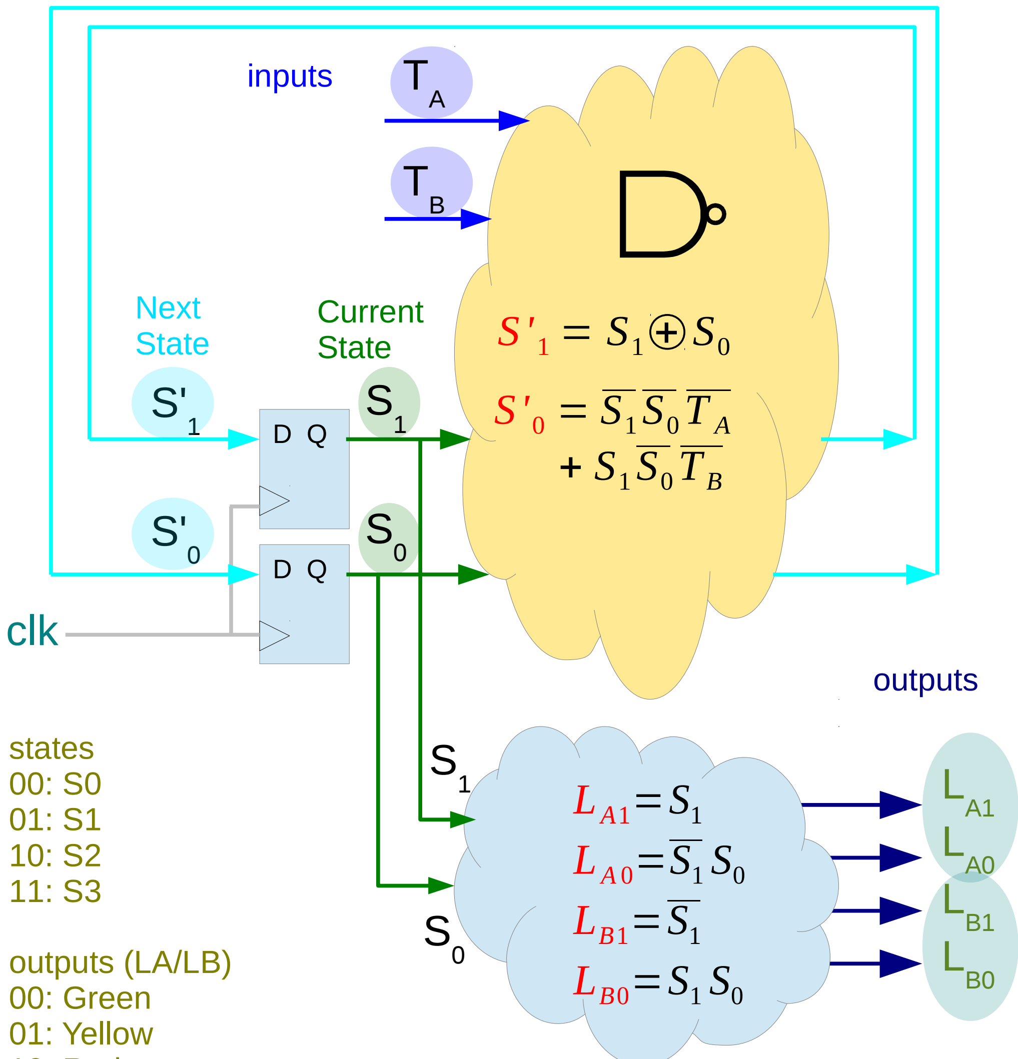


outputs (LA/LB)
00: Green
01: Yellow
10: Red
11: X

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

Young Won Lim
6/9/18

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

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

$$S'_1 = S_1 + S_0$$

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

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

Current State		**FSM Inputs**		**Next State**
$(S_1 S_0)$		$(T_A T_B)$		$(S_1 S_0)$
$\{00, 01, 10, 11\}$	$\times$	$\{00, 01, 10, 11\}$	➔	$\{00, 01, 10, 11\}$

Cartesian Product

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	S'_0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	0
1	0	1	0	1	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

Current State $(S_1 S_0)$

FSM Inputs $(T_A T_B)$

Next State $(S_1 S_0)$

$$\{00, 01, 10, 11\} \times \{00, 01, 10, 11\} \rightarrow \{00, 01, 10, 11\}$$

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

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

G : 00
Y : 01
R : 10

$$L_{A1} = S_1$$
$$L_{A0} = \overline{S_1} S_0$$
$$L_{B1} = \overline{S_1}$$
$$L_{B0} = S_1 S_0$$

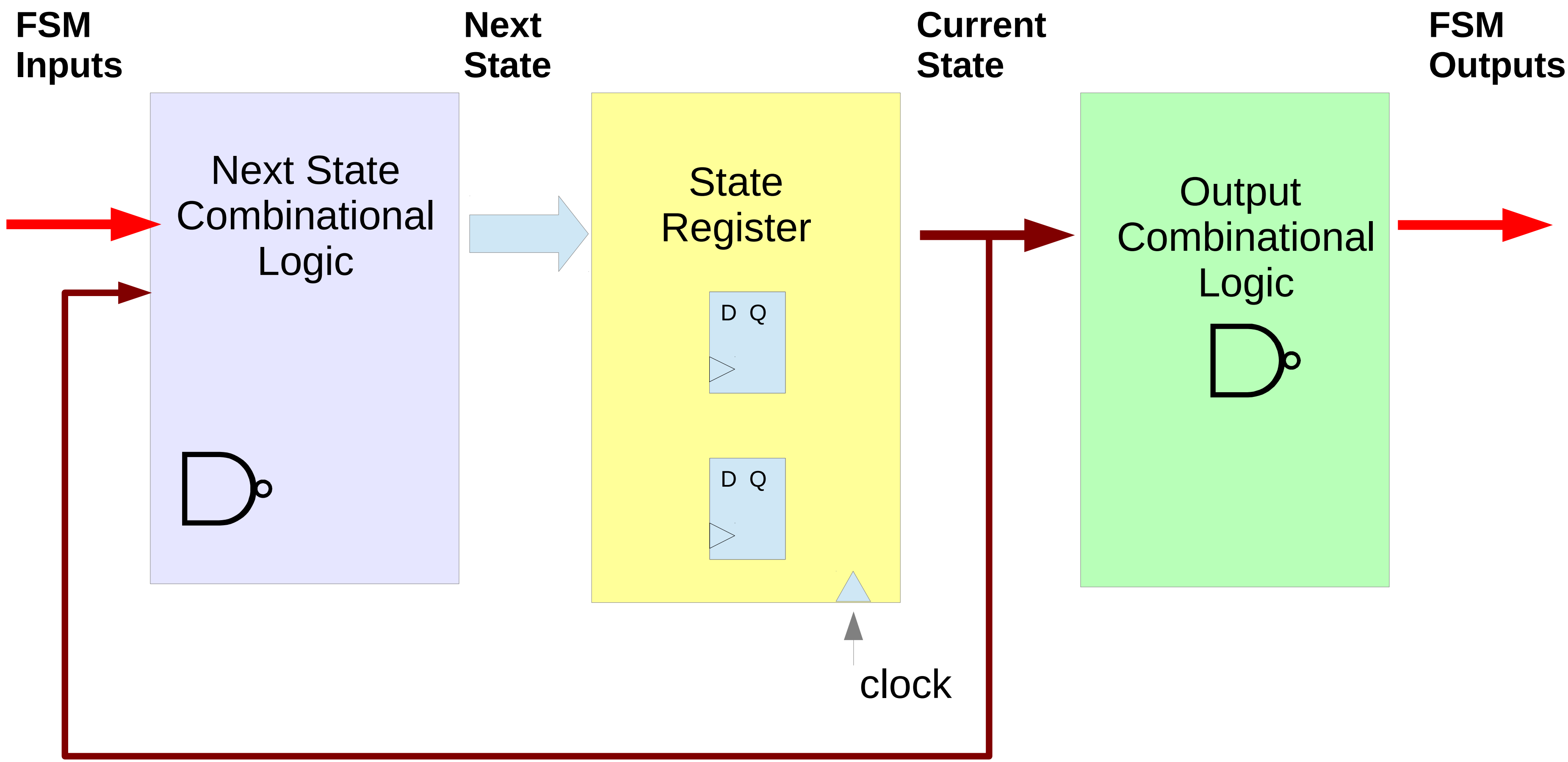
https://en.wikiversity.org/wiki/The_necessities_in_Computer_Design

Current State $(S_1 S_0)$

FSM Output $(L_{A1,} L_{A0,} L_{B1,} L_{B0})$

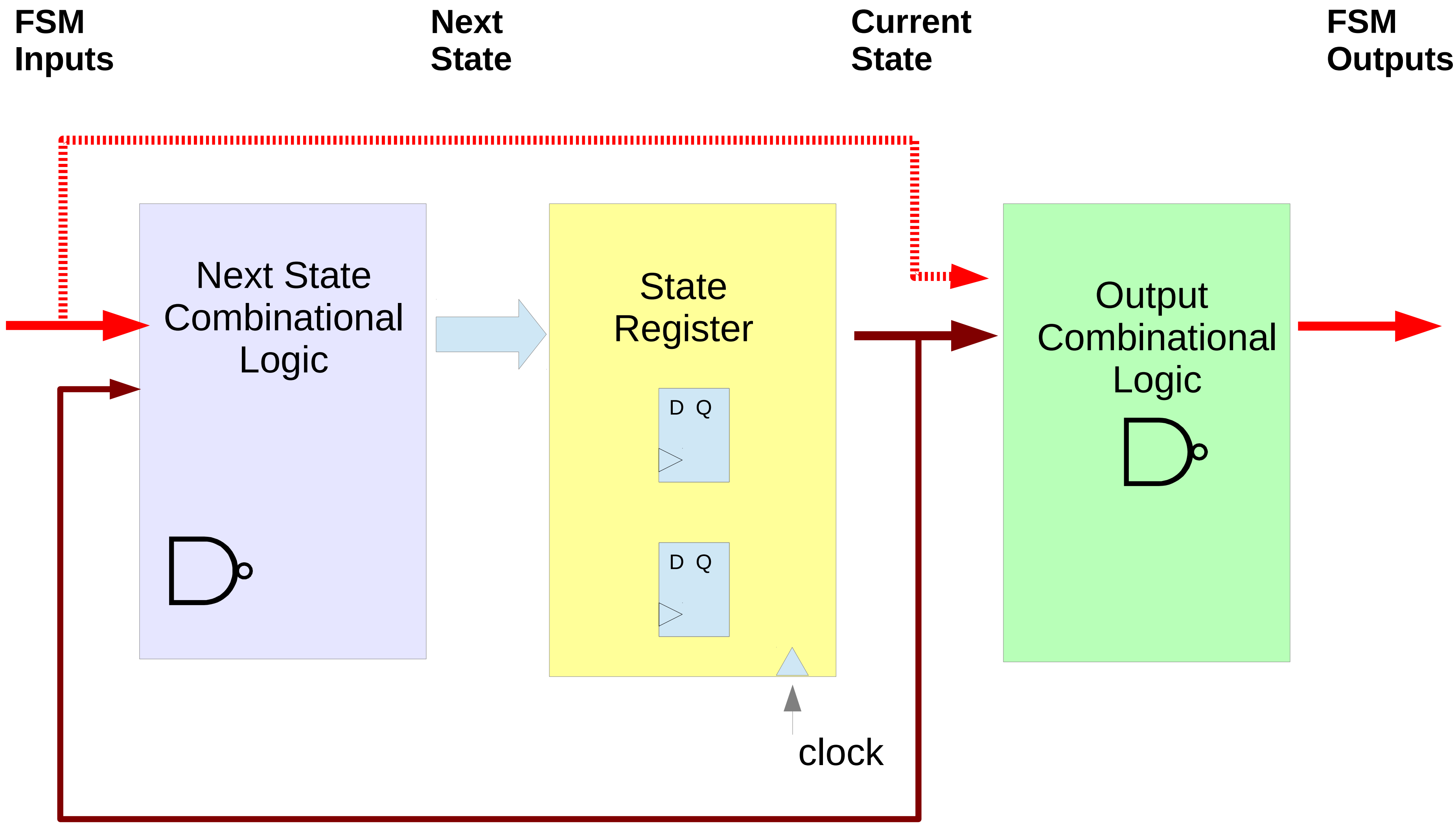
$\{00, 01, 10, 11\}$ → $\{0010, 0110, 1000, 1001\}$

Moore FSM



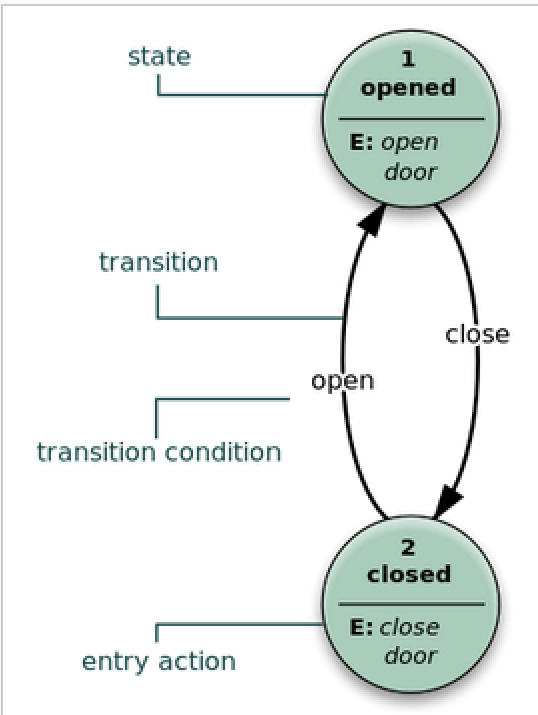
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

Mealy FSM

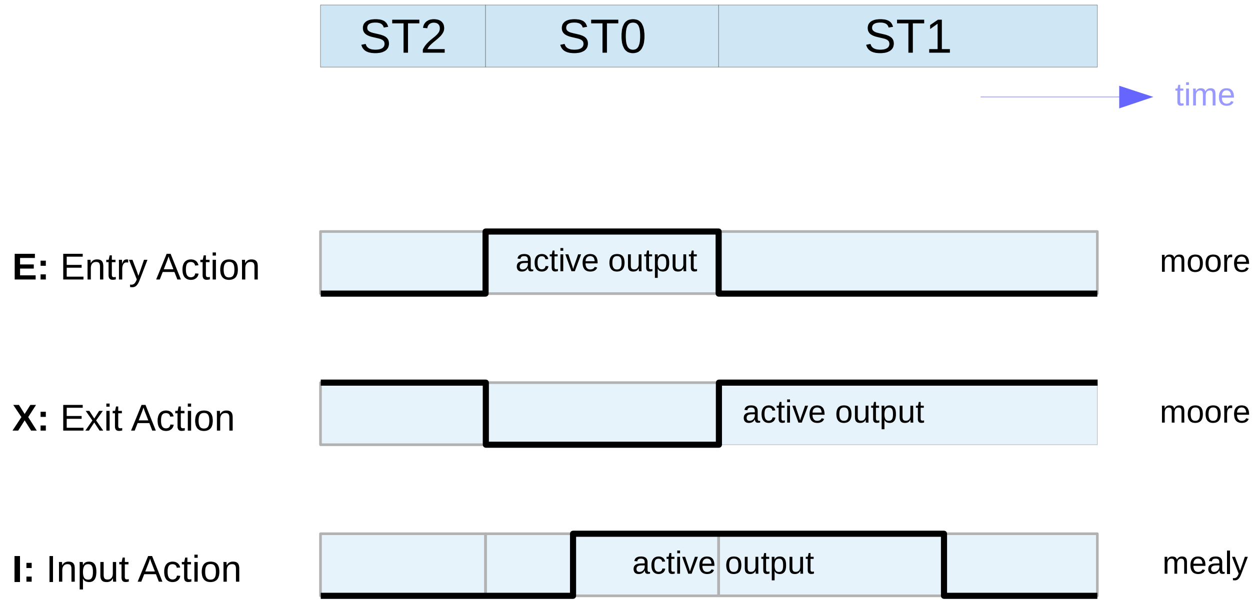


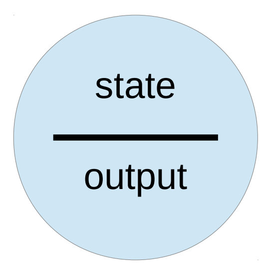
https://en.wikiversity.org/wiki/The_necessities_in_Digital_Design

State Diagram



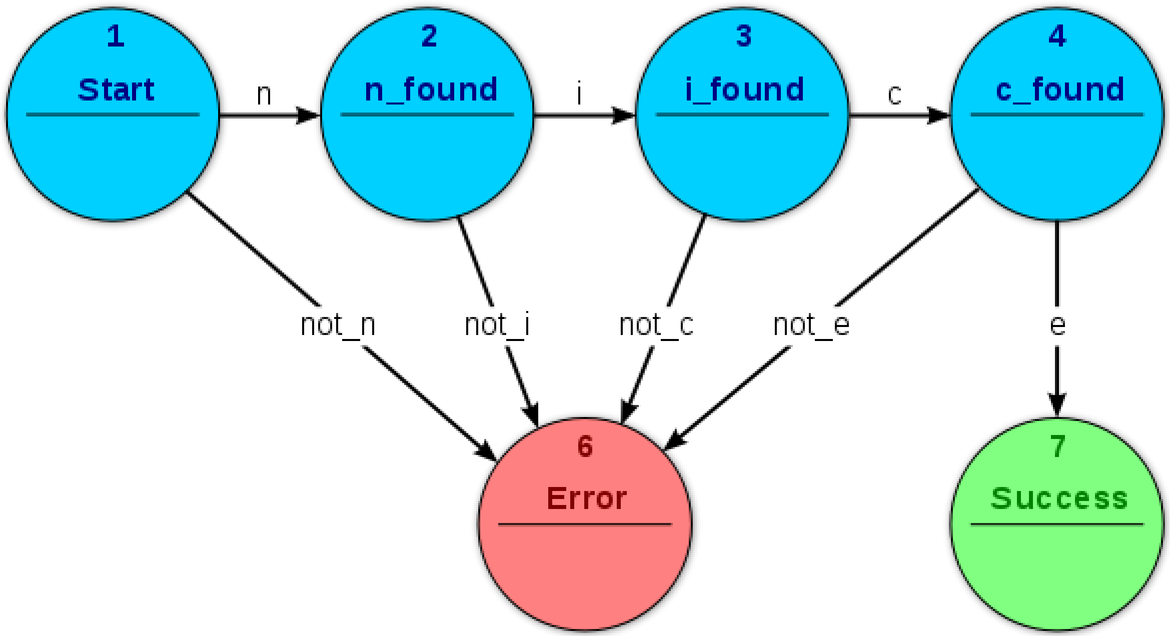
A state diagram for a door that can only be opened and closed





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

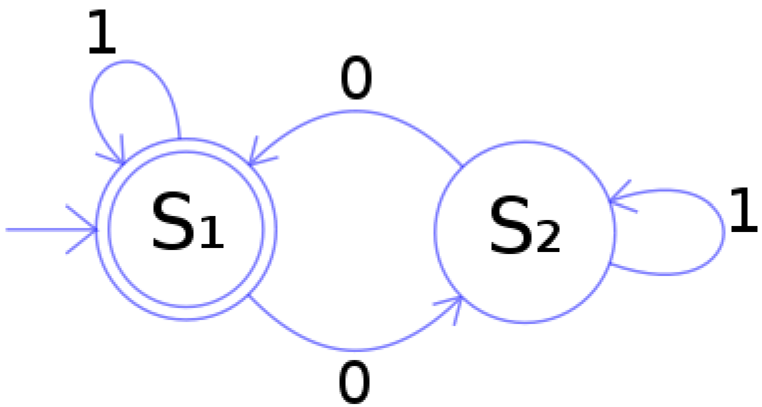
Acceptors



Acceptor FSM: parsing the string "nice"

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

Recognizers



Representation of a finite-state machine;
determines whether a binary number has
an **even** number of **0s**,
where $\mathbf{S_1}$ is an **accepting state**.

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

Classifiers

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

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

Transducers

Transducers generate **output** based on a given **input** and/or a **state** using actions.

They are used for control applications and in the field of computational linguistics.

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

Acceptors, Recognizers, Transducers

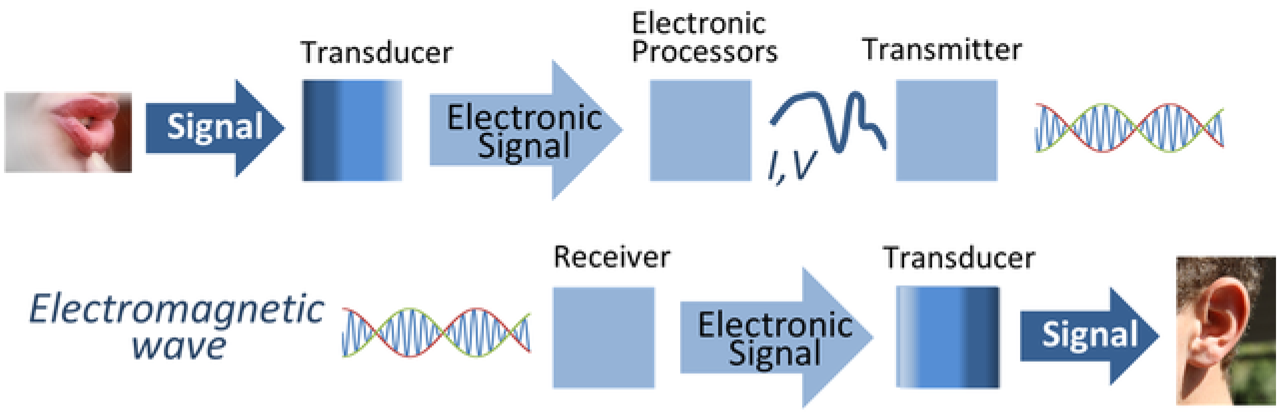
acceptors: either accept the input or not

recognizers: either recognize the input

transducers: generate output from given input

https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/automata-theory/basics.html

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

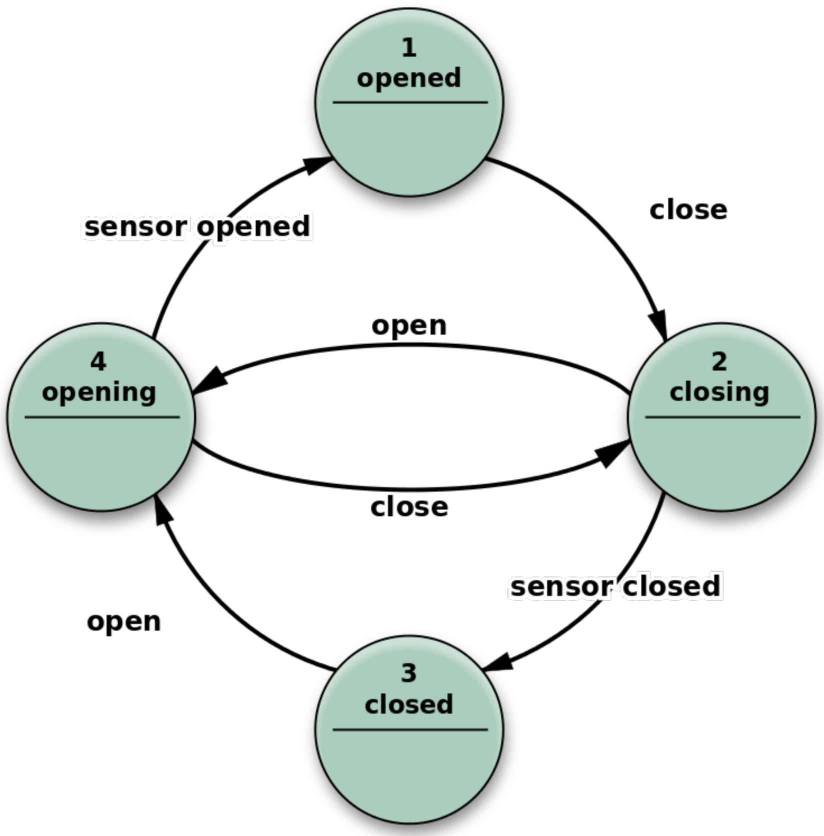


Fig. 6 Transducer FSM: Moore model example

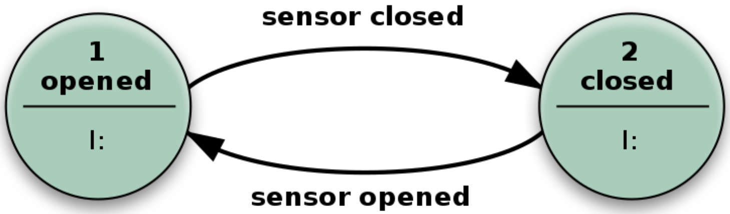


Fig. 7 Transducer FSM: Mealy model example

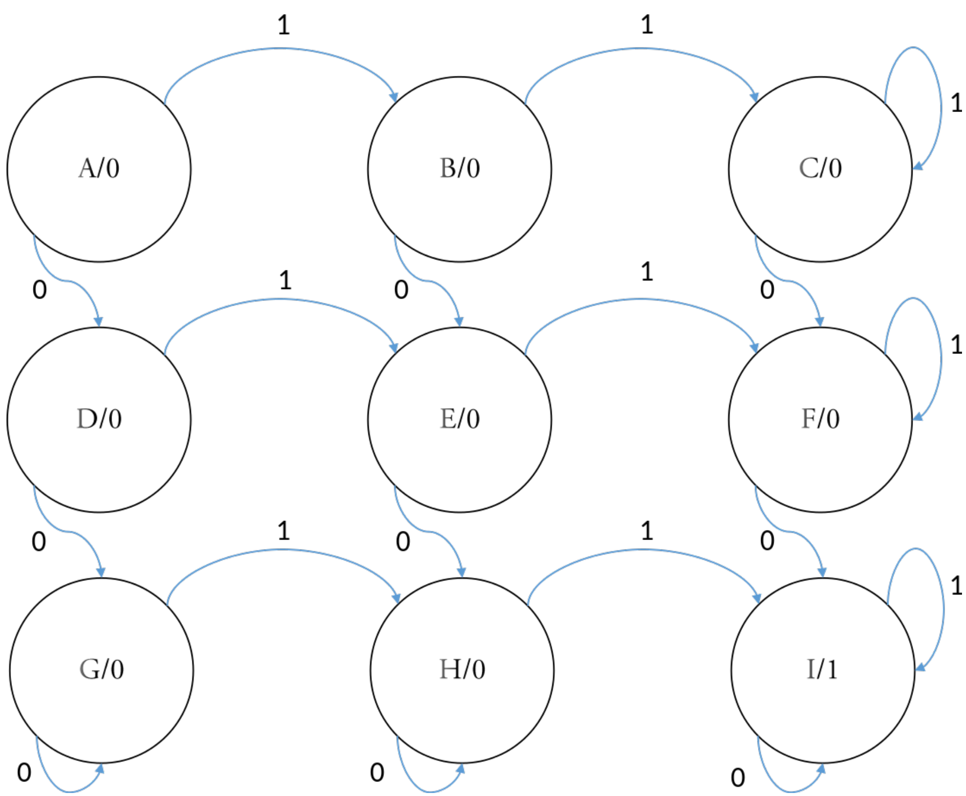
There are two **input actions** (I:):

"start motor to close the door
if command_close arrives"

"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

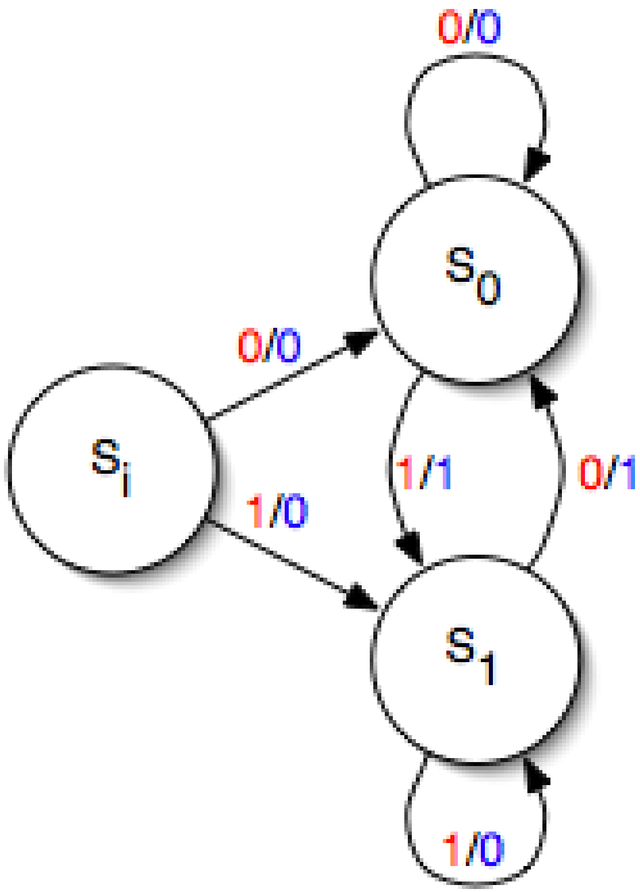


Current state	Input	Next state	Output
A	0	D	0
	1	B	
B	0	E	0
	1	C	
C	0	F	0
	1	C	
D	0	G	0
	1	E	
E	0	H	0
	1	F	
F	0	I	0
	1	F	
G	0	G	0
	1	H	
H	0	H	0
	1	I	
I	0	I	1
	1	I	

output does not depend on **inputs**

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

Mealy machine



input / output

output does depend on **inputs**

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

Mathematical Model – transducers (1)

A **finite-state transducer** is a sextuple (**Σ**, **Γ**, **S**, $\mathbf{s_0}$, **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**.
- $\mathbf{s_0}$ is the **initial** state, an element of S.
- **δ** is the **state-transition function**: $\boldsymbol{\delta : S \times \Sigma \rightarrow S}$
- **ω** is the **output function**.

  Moore machine : $\boldsymbol{\omega : S \rightarrow \Gamma}$

  Mealy machine : $\boldsymbol{\omega : S \times \Sigma \rightarrow \Gamma}$

(Σ, Γ, S, s_0, δ, ω)

(I, O, S, f, g, σ)

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

Mathematical Model – transducers (2)

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

If the **output** function depends only on a **state** ($\boldsymbol{\omega : S \rightarrow \Gamma}$) 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

Mathematical Models – acceptors

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

output set {0, 1}

- **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**: **δ : S × Σ → S**
- **F** is the set of **final states**, a (possibly empty) subset of S.

output function **ω**
A set of accepted states

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

Finite State Tranducers and Acceptors

limited finite state machine (Σ, S, δ)

finite state acceptor $(\Sigma, S, s_0, \delta, F)$ ← **Finite State Automaton (FSA)**

finite-state transducer $(\Sigma, \Gamma, S, s_0, \delta, \omega)$ ← **Finite State Machine (FSM)**

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

References

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