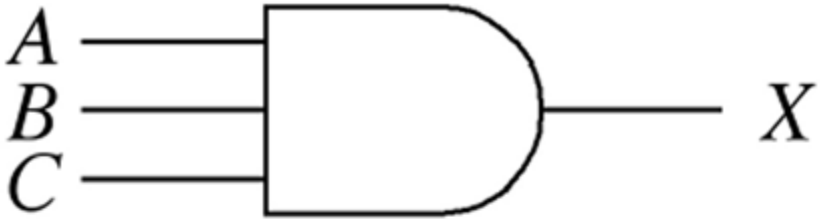
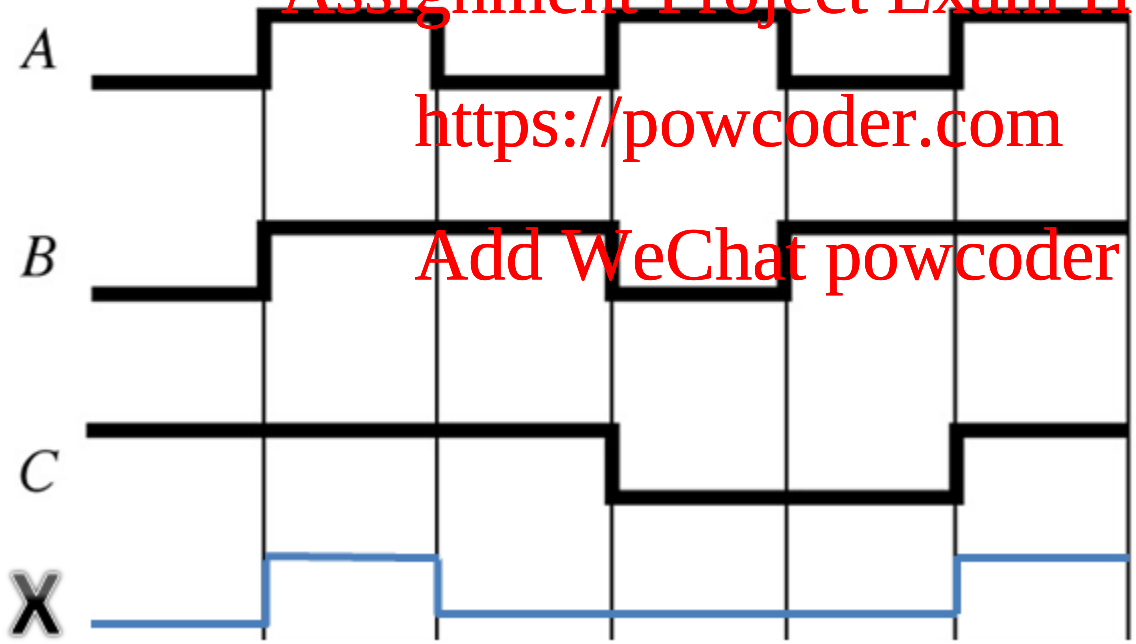
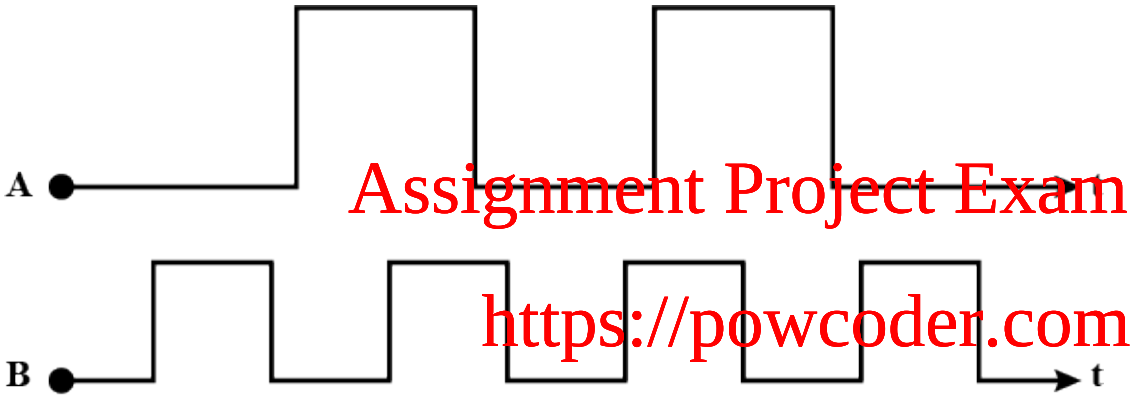
On wave diagrams





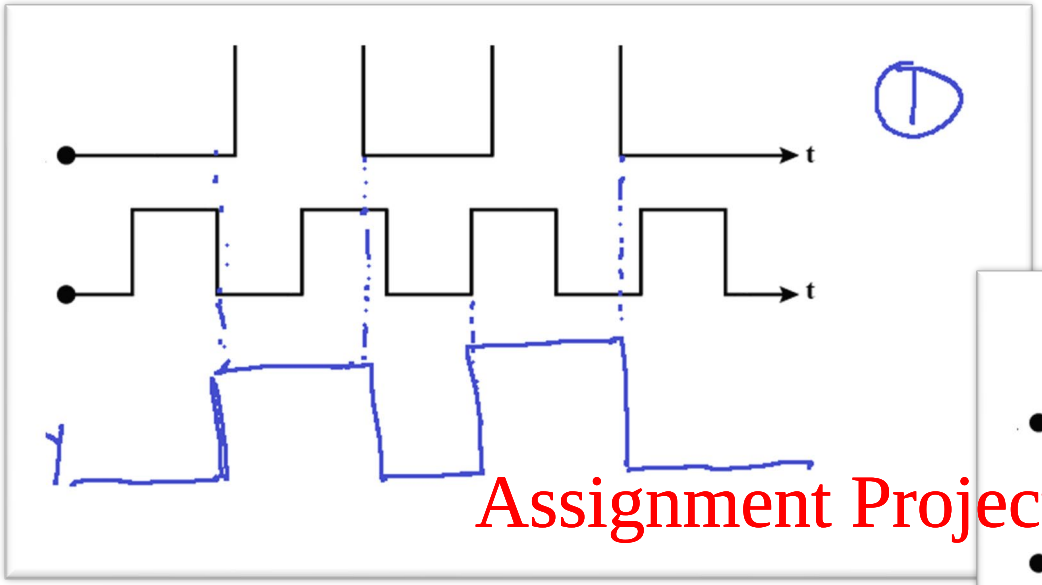
On wave diagrams

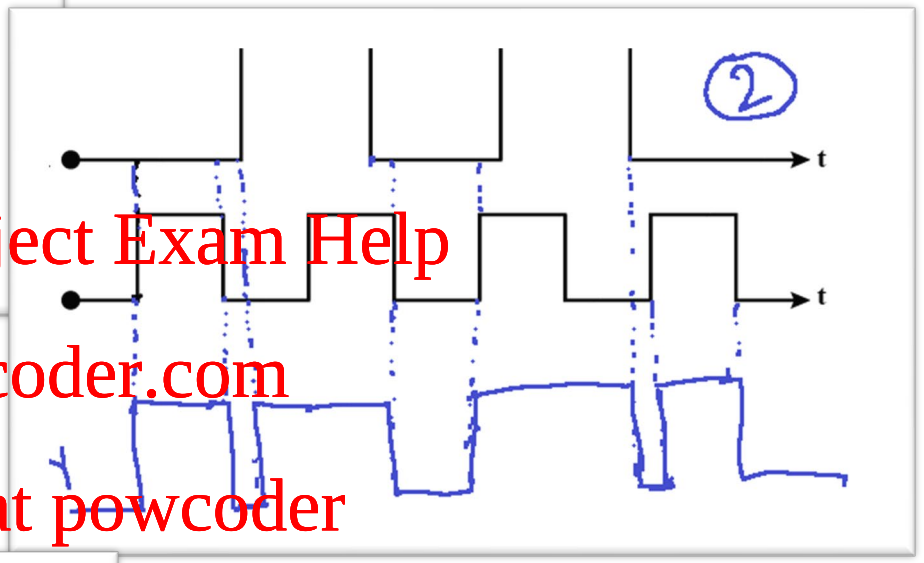
Y=A OR B

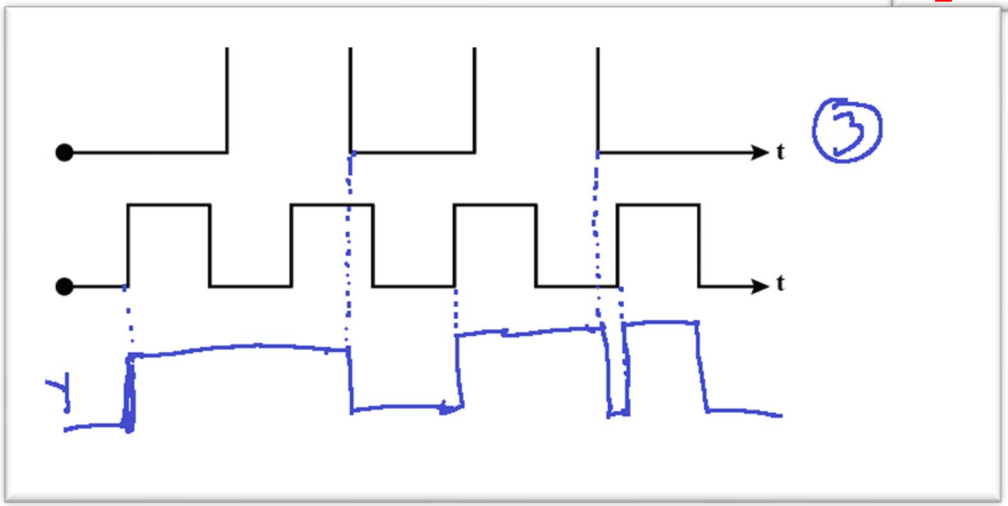


What will be the waveform of output Y

(Class Poll)







Sequential Circuits and Memory

Combinational vs. Sequential

- Combinational circuit
  - Always gives the same output for a given set of inputs
  - Example: Adder always generates sum and carry, regardless of previous inputs
- Sequential circuit
  - **Remembers** previous input
  - Output depends on state and input

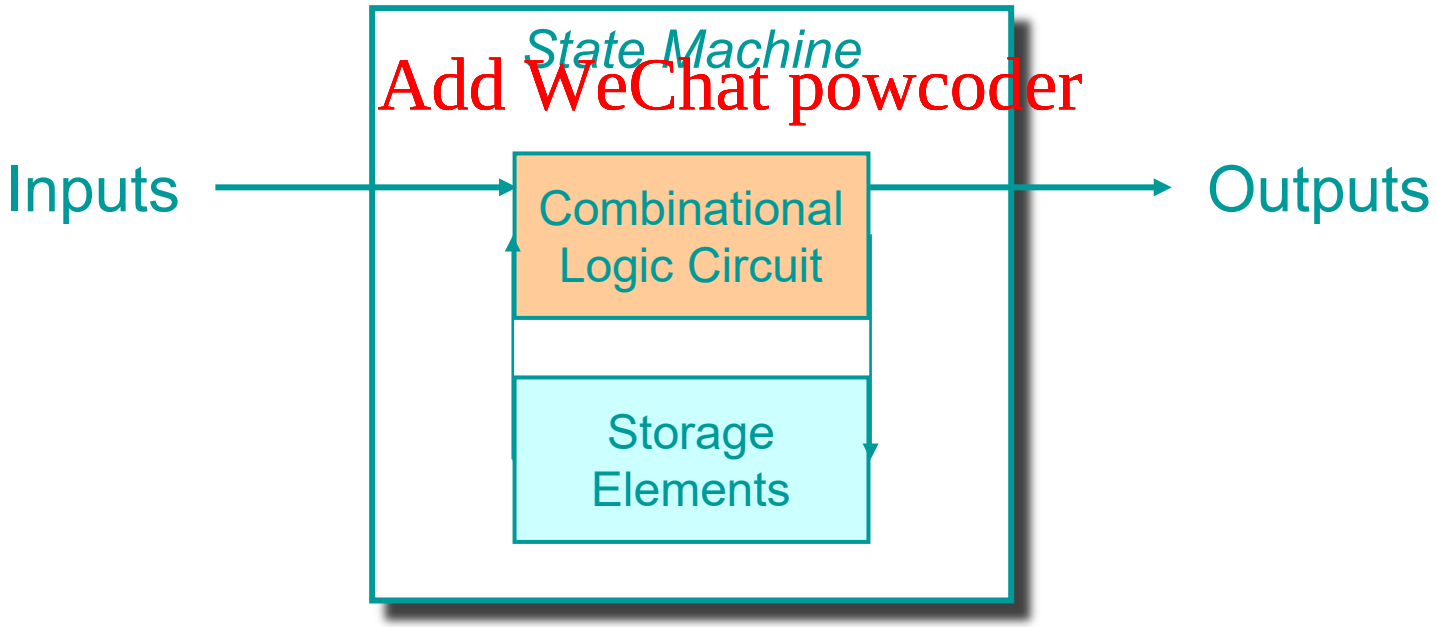
Sequential Circuits

- **Store** information
- Output depends on stored information (state) plus input
  - So a given input might produce different outputs, depending on the stored information
- Example: ticket counter
  - Advances when you push the button
  - Output depends on previous state

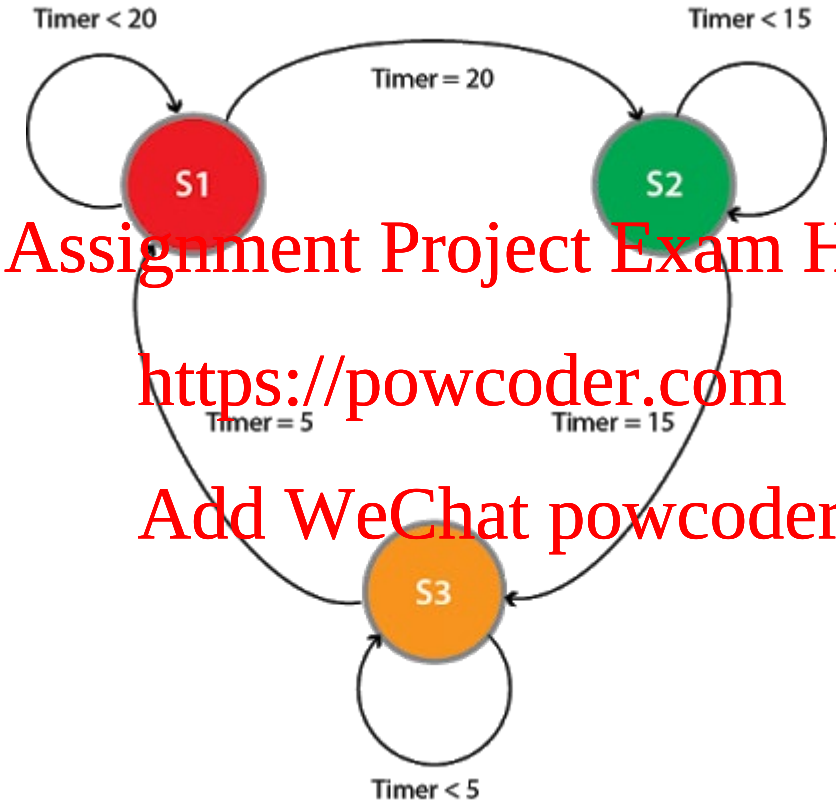
State Machine

The basic type of sequential circuit

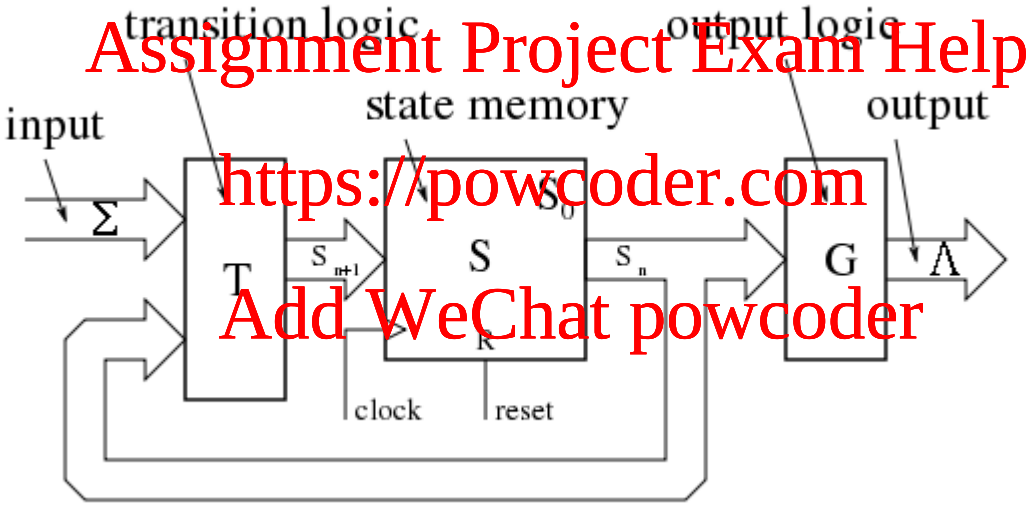
- Combines combinational logic with storage
- "Remembers" state, and changes output (and state) based on inputs and current state



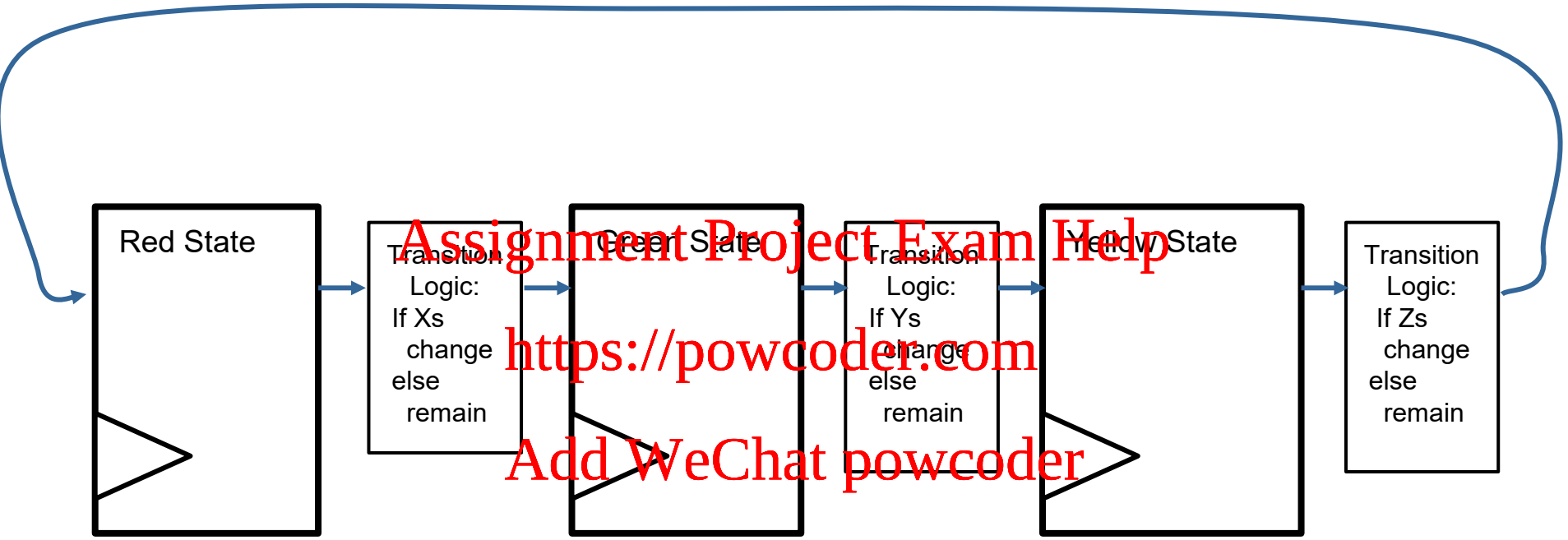
Example of State Machine: Traffic Light



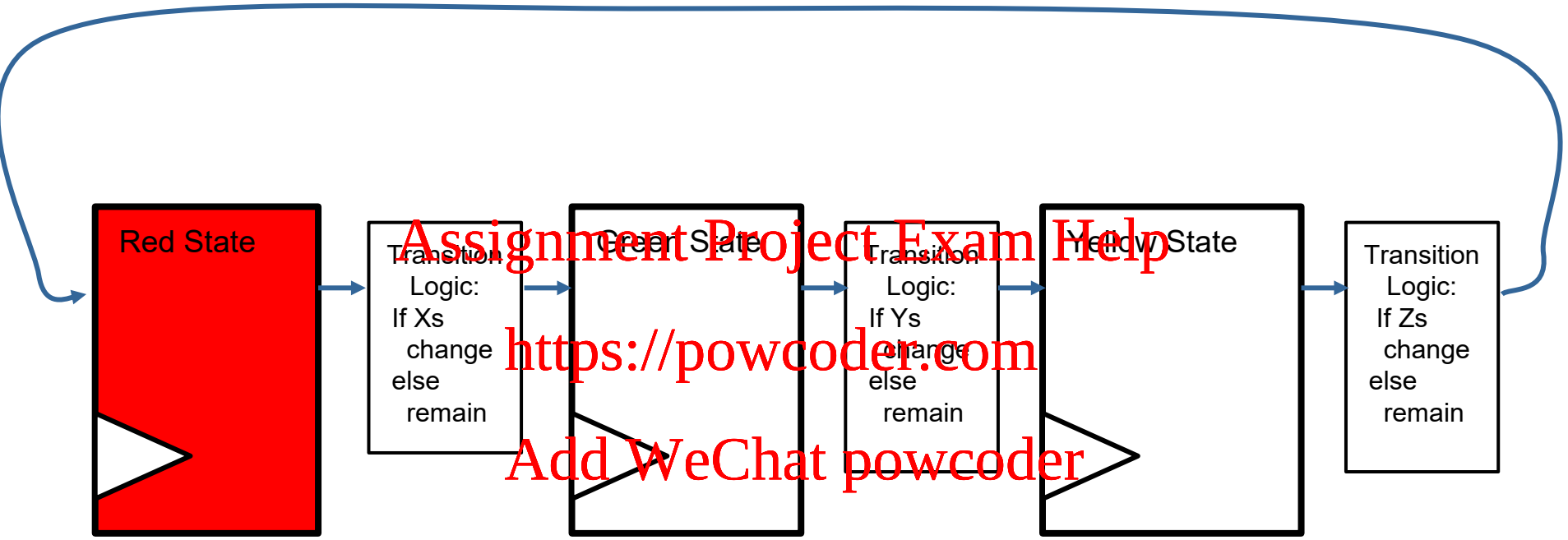
State Machine: Overview



State Machine: Traffic Light



State Machine: Traffic Light

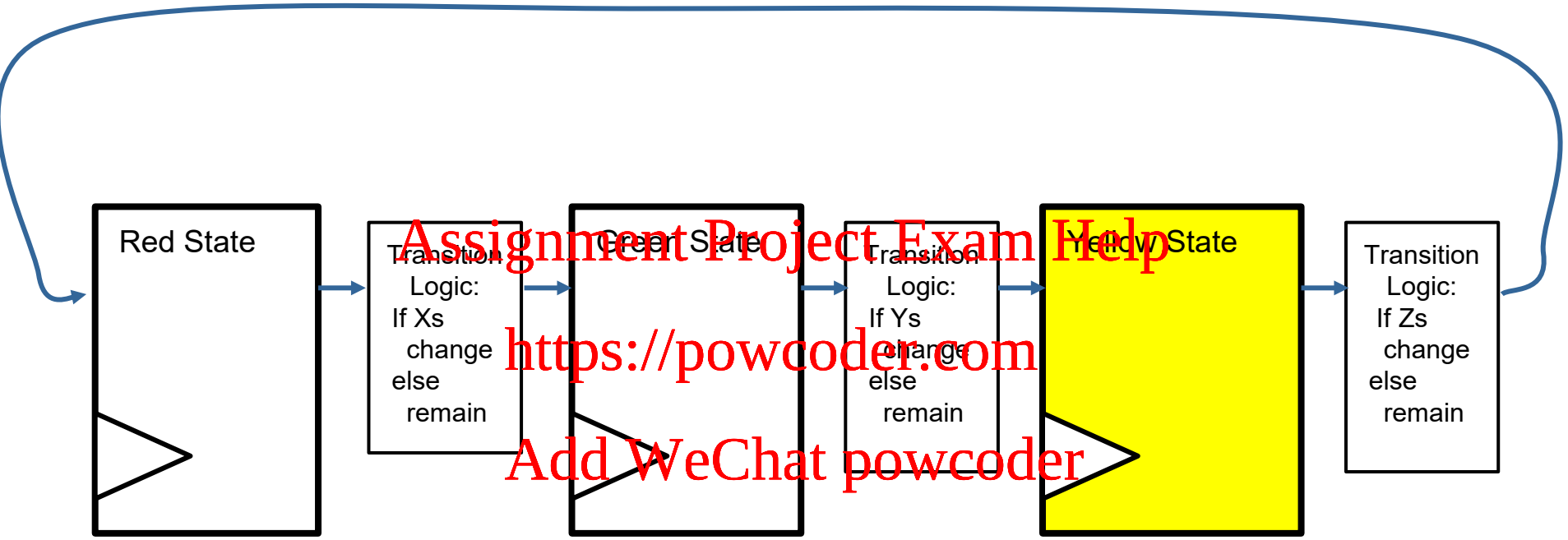


- Start off in default state Red

State Machine: Traffic Light

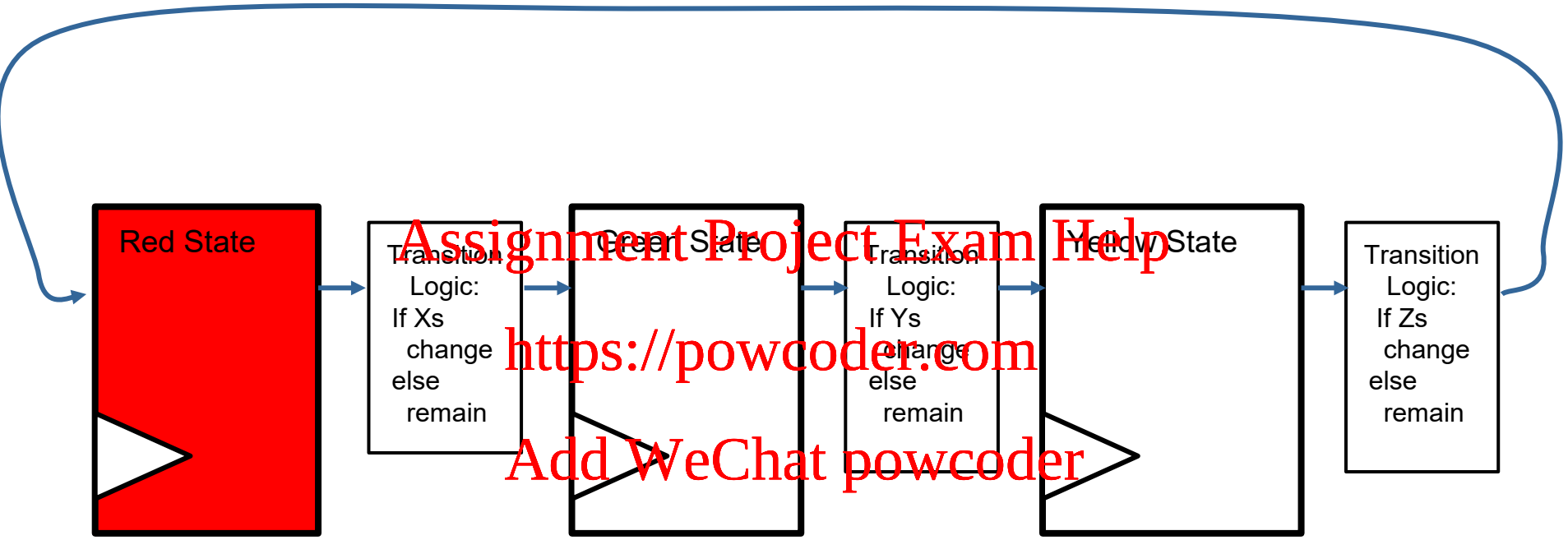
- After a specified time Xs switch to next state

State Machine: Traffic Light



- After a specified time Ys switch to next state

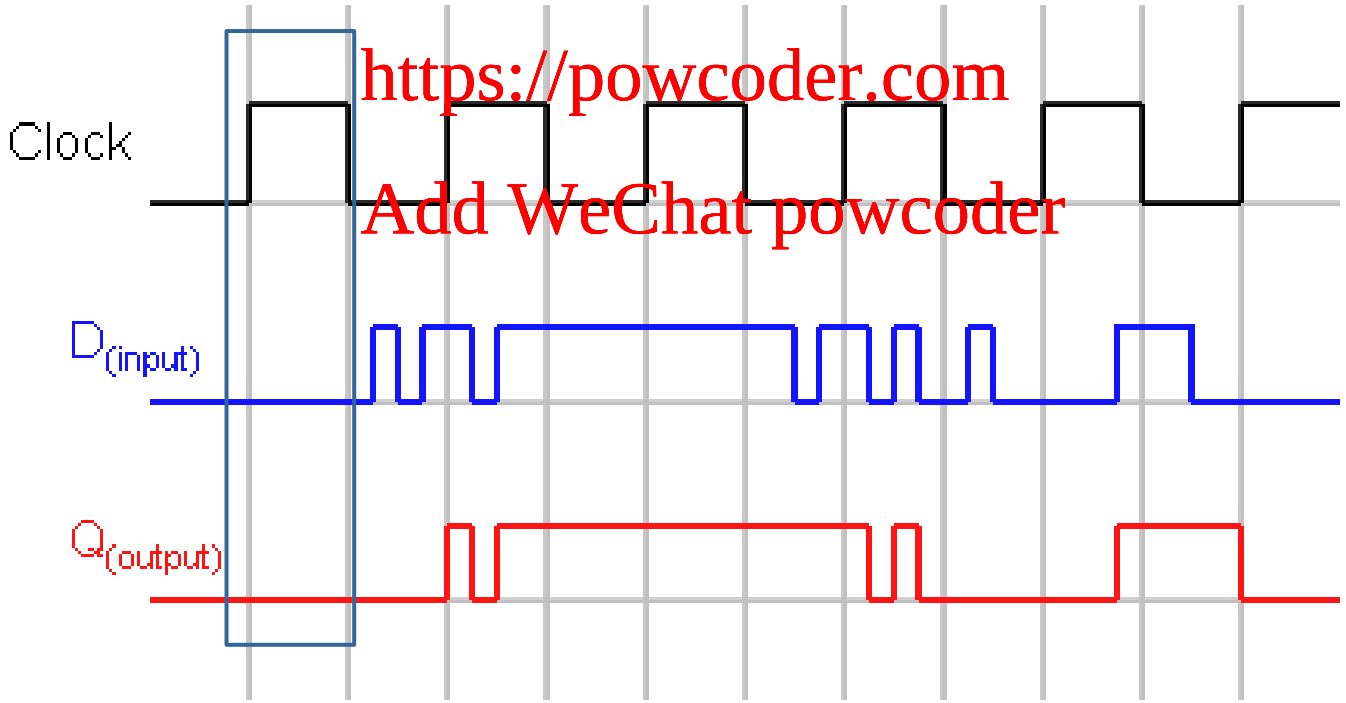
State Machine: Traffic Light



- After a specified time Zs switch to next state which is Red.

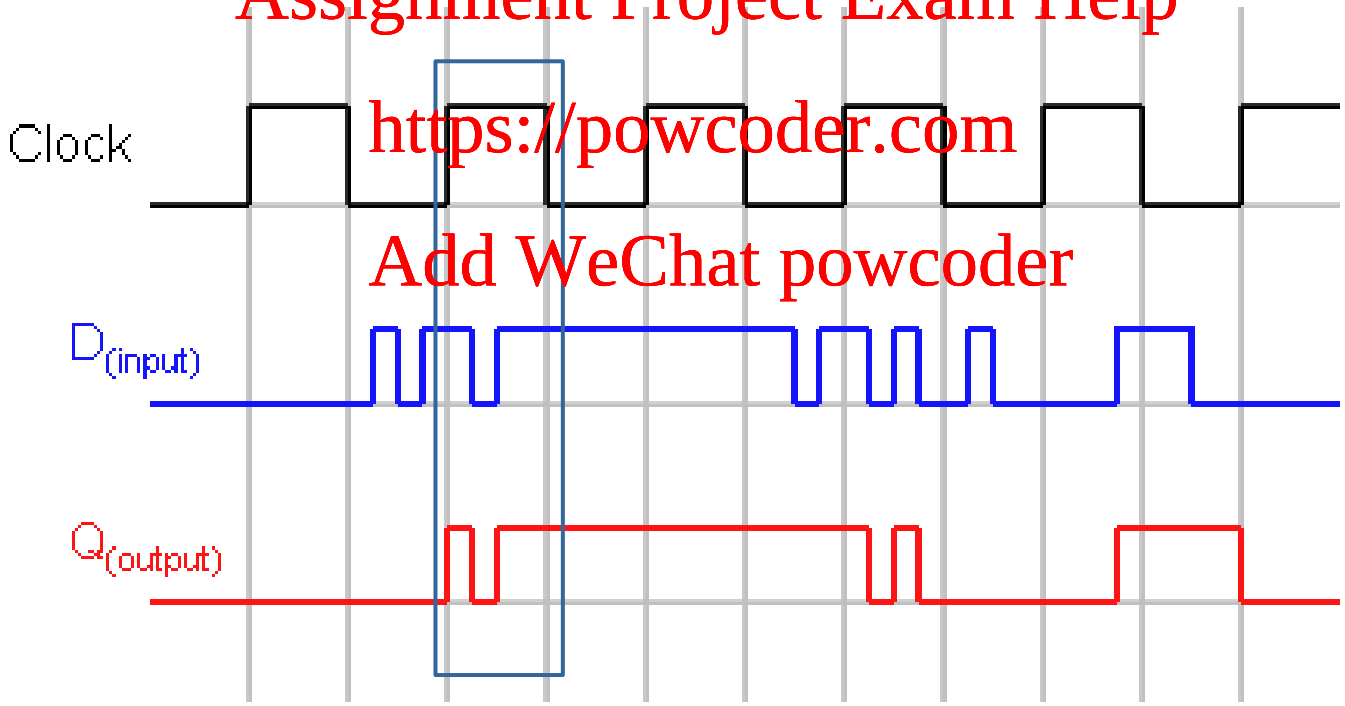
Latch

- Output is equal to Input when clk is high.
- Level sensitive
- Stores last value when clk is low.



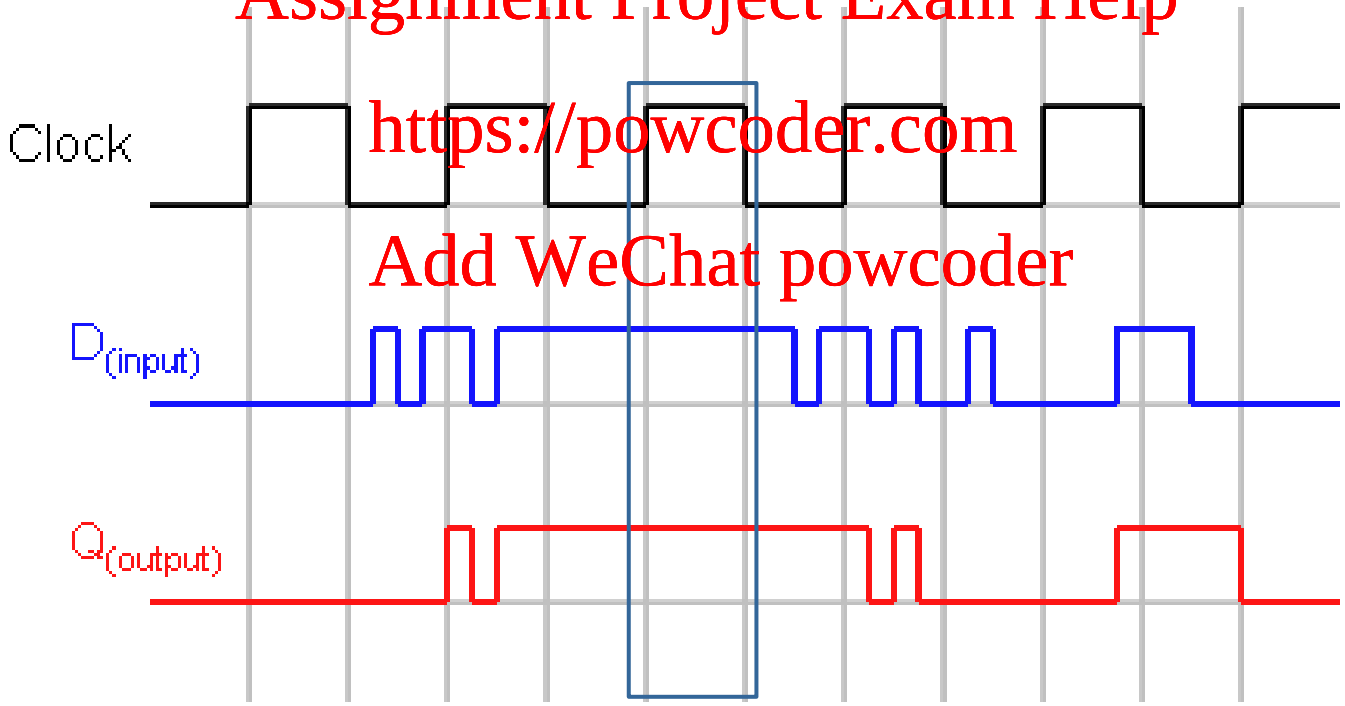
Latch

- Output is equal to Input when clk is high.
- Stores last value when clk is low.



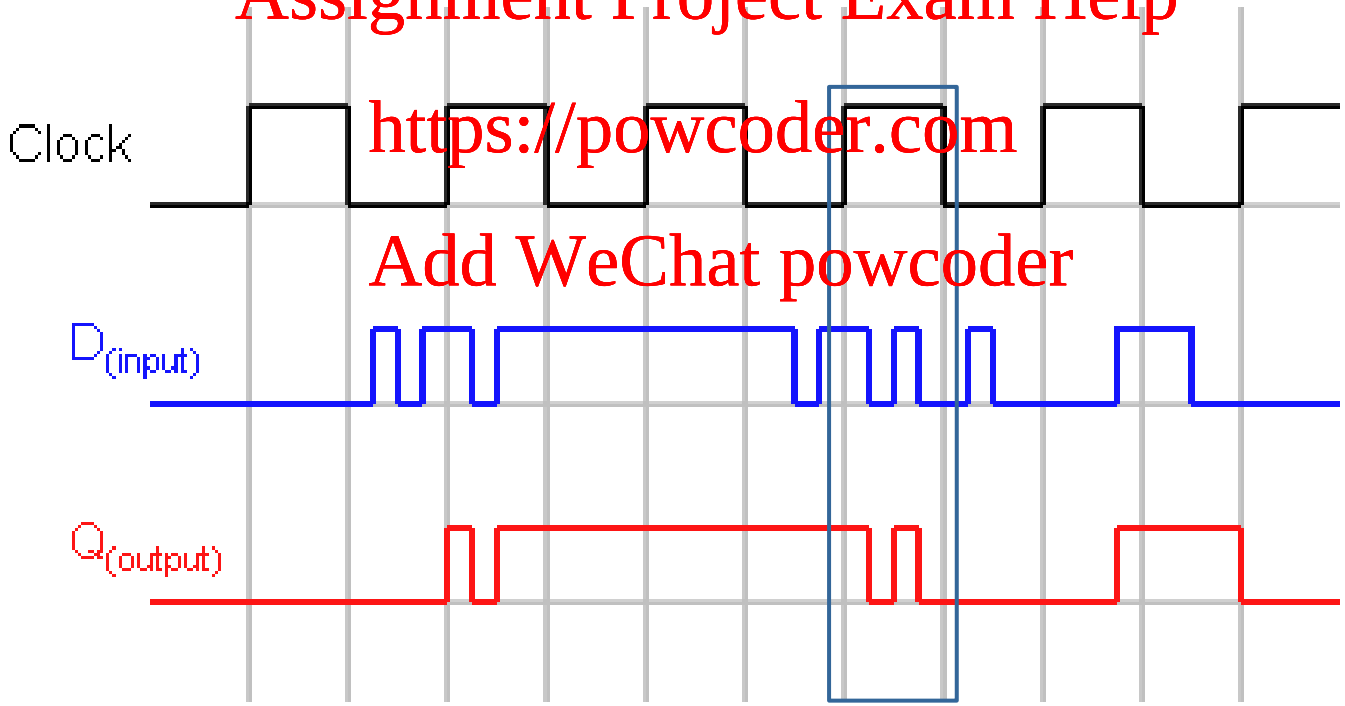
Latch

- Output is equal to Input when clk is high.
- Stores last value when clk is low.



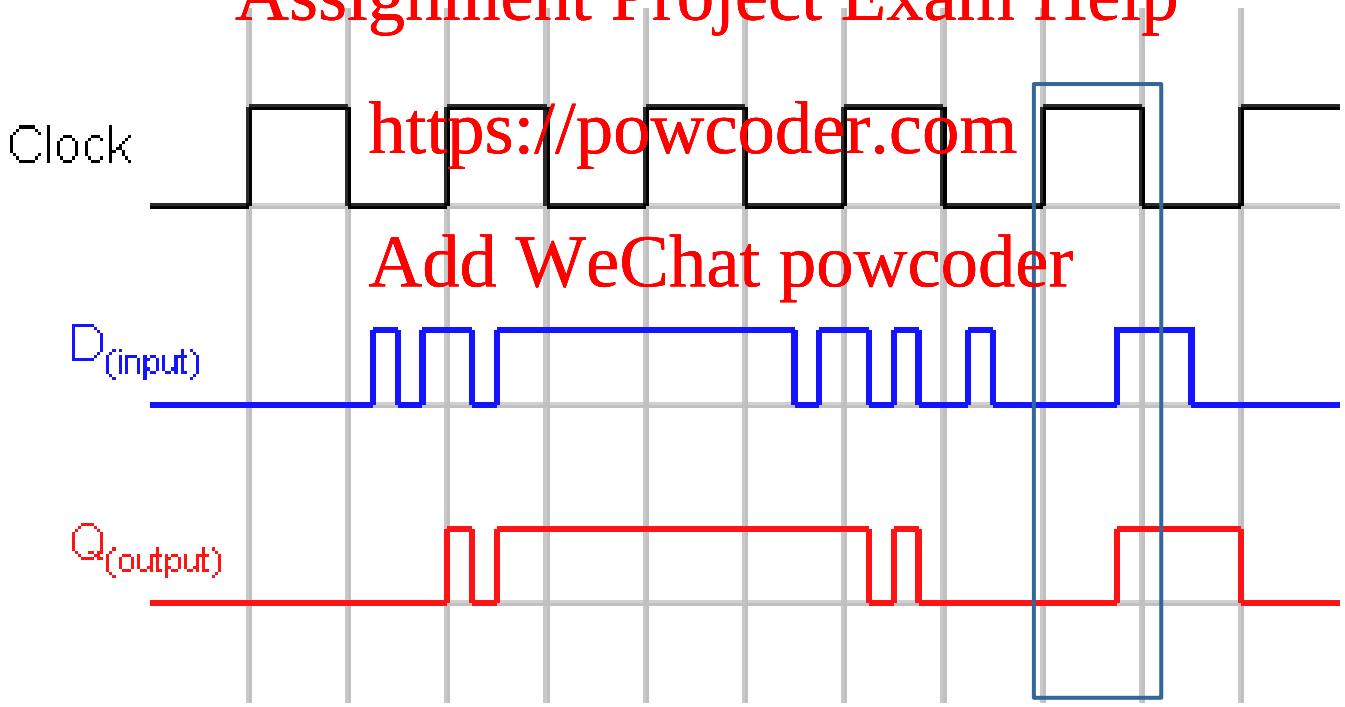
Latch

- Output is equal to Input when clk is high.
- Stores last value when clk is low.



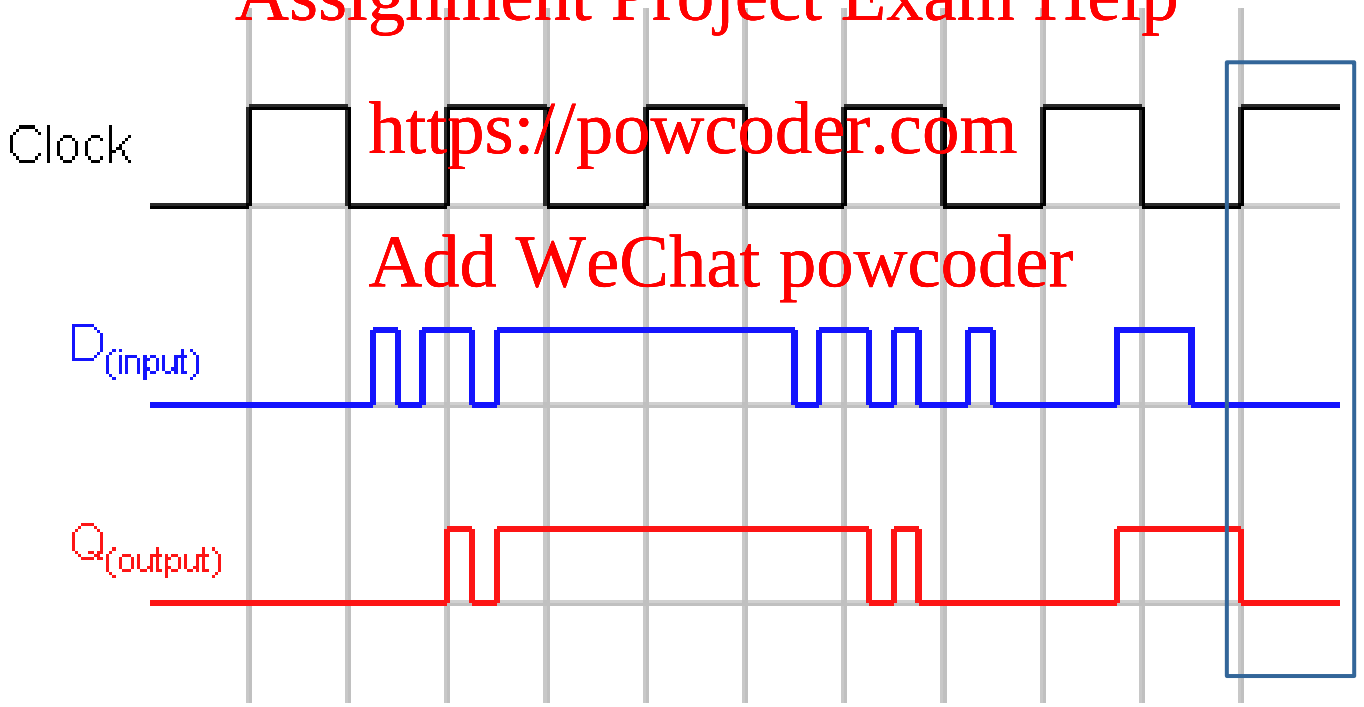
Latch

- Output is equal to Input when clk is high.
- Stores last value when clk is low.



Latch

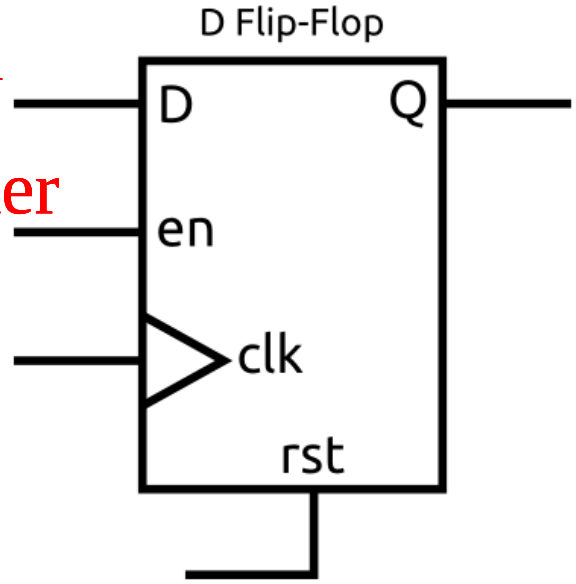
- Output is equal to Input when clk is high.
- Stores last value when clk is low.



D Flip-Flops

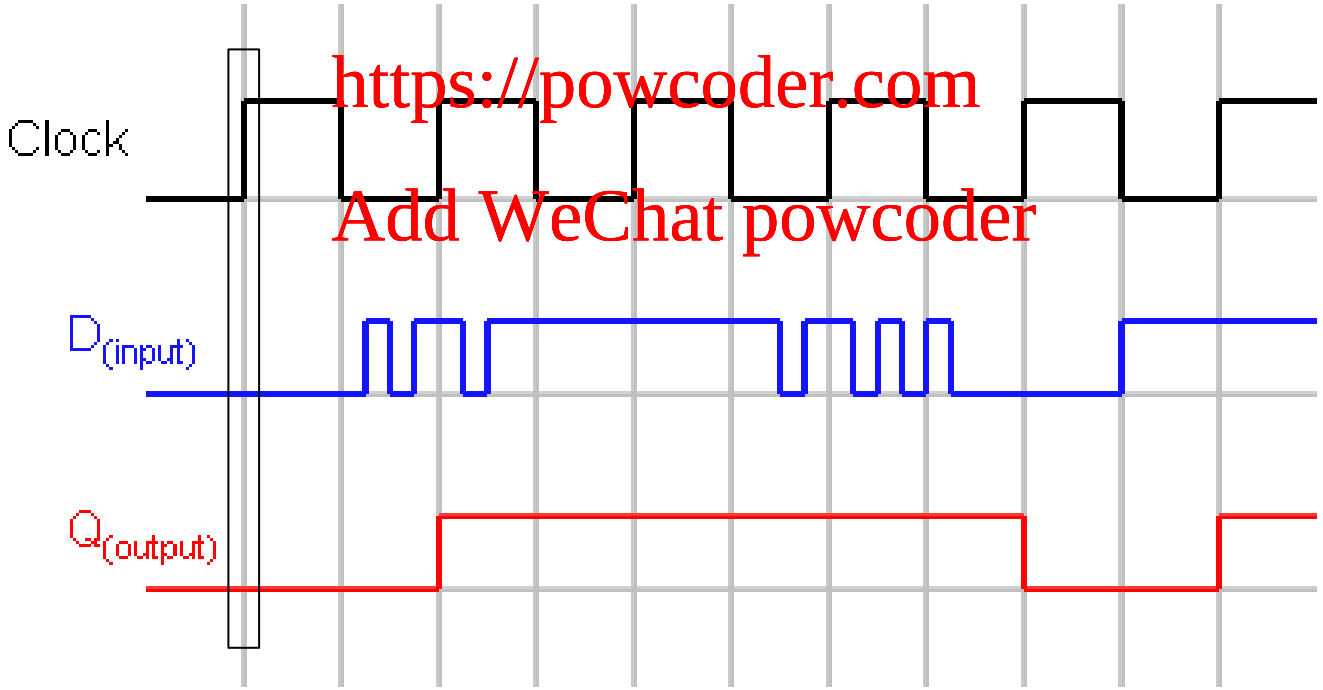
Memory device

- Can be positive **edge** triggered or negative **edge** triggered (by a clock usually abbreviated by clk)
- Different types: RS, JK
- Inputs/Outputs:
  - D: input signal
  - clk: Clock signal
  - en: if 0 Q holds its value, if 1, Q becomes D at clk edge.
  - rst: if 1 then Q becomes 0
  - Q: output signal



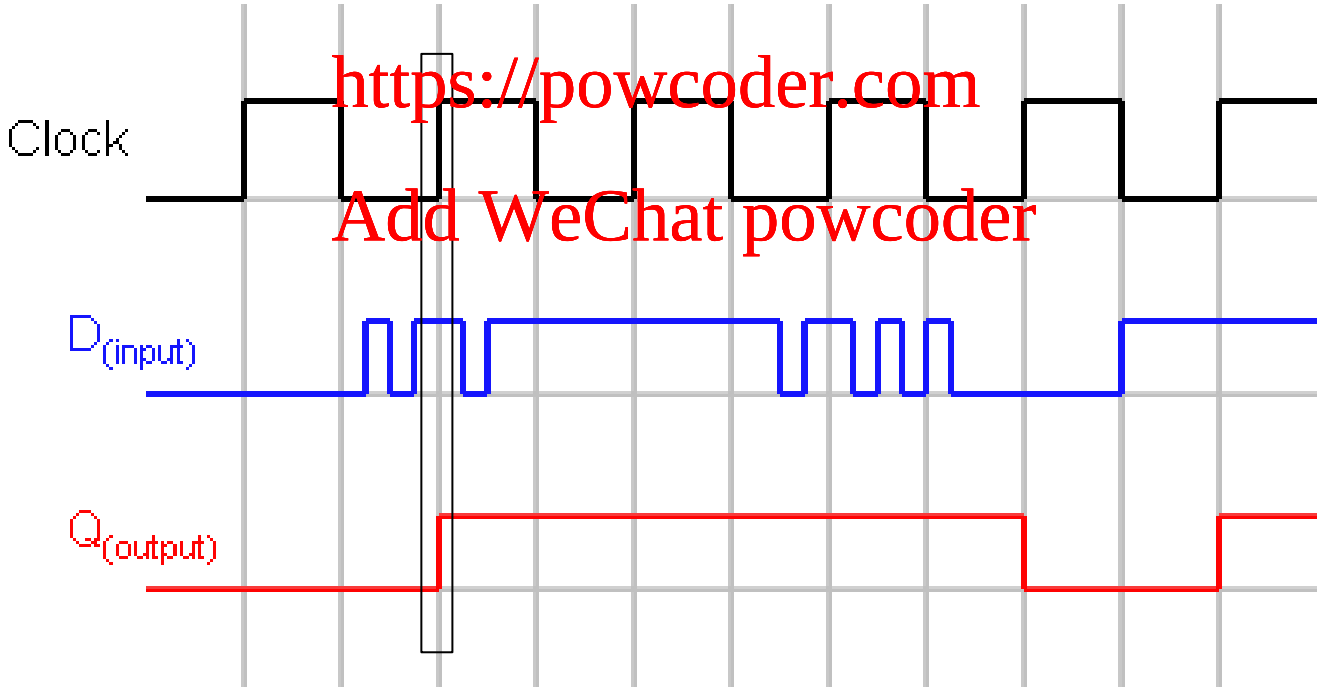
D flip-flop(positive-edge) timing diagram

- Q becomes D at positive clk edge (0 -> 1).
  - Stores value until next positive clk edge.
- clk oscillates between 0 and 1
  - frequency = 1/period



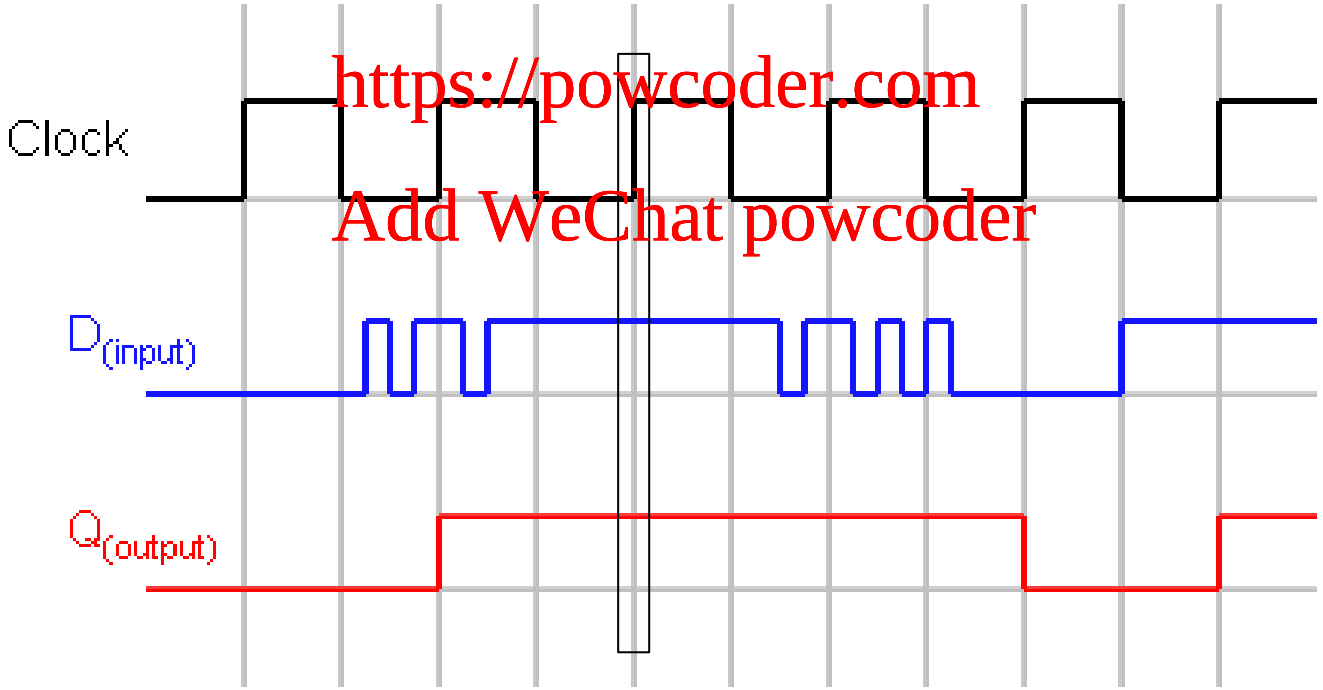
D flip-flop(positive-edge) timing diagram

- Q becomes D at positive clk edge (0 -> 1).
  - Stores value until next positive clk edge.
- clk oscillates between 0 and 1
  - frequency = 1/period



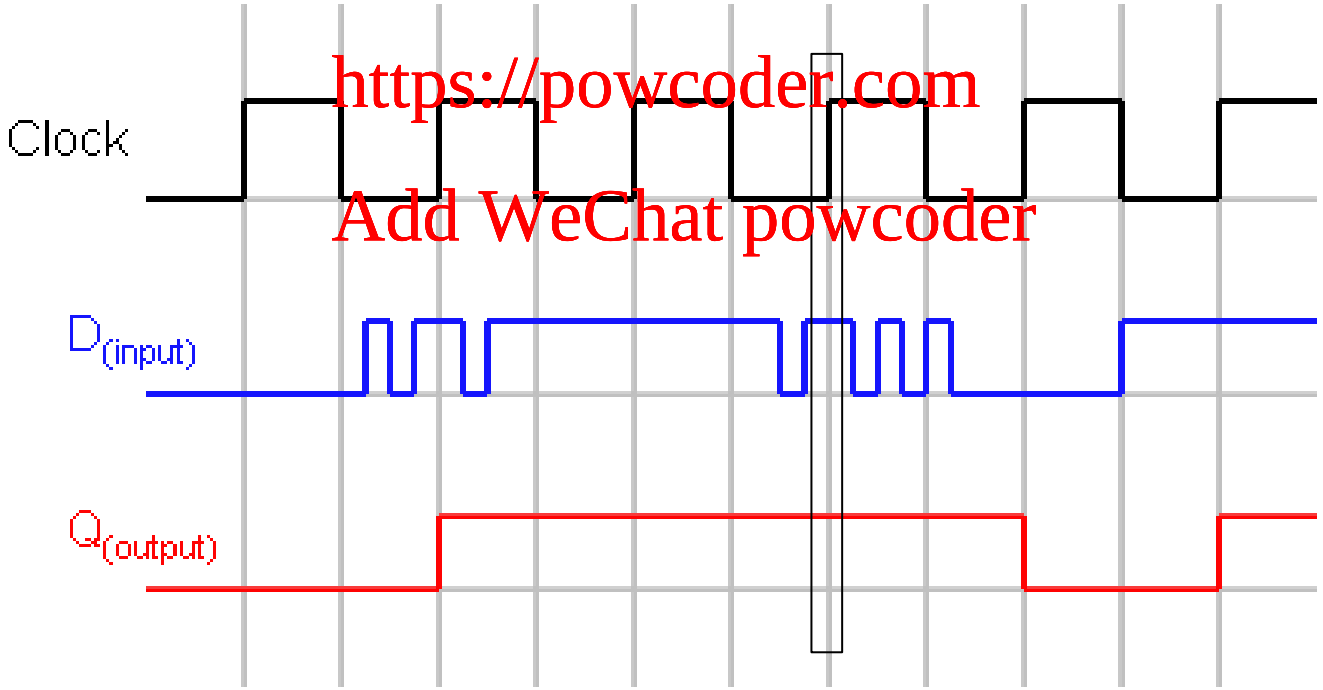
D flip-flop(positive-edge) timing diagram

- Q becomes D at positive clk edge.
  - Stores value until next positive clk edge.
- clk oscillates between 0 and 1
  - frequency = 1/period



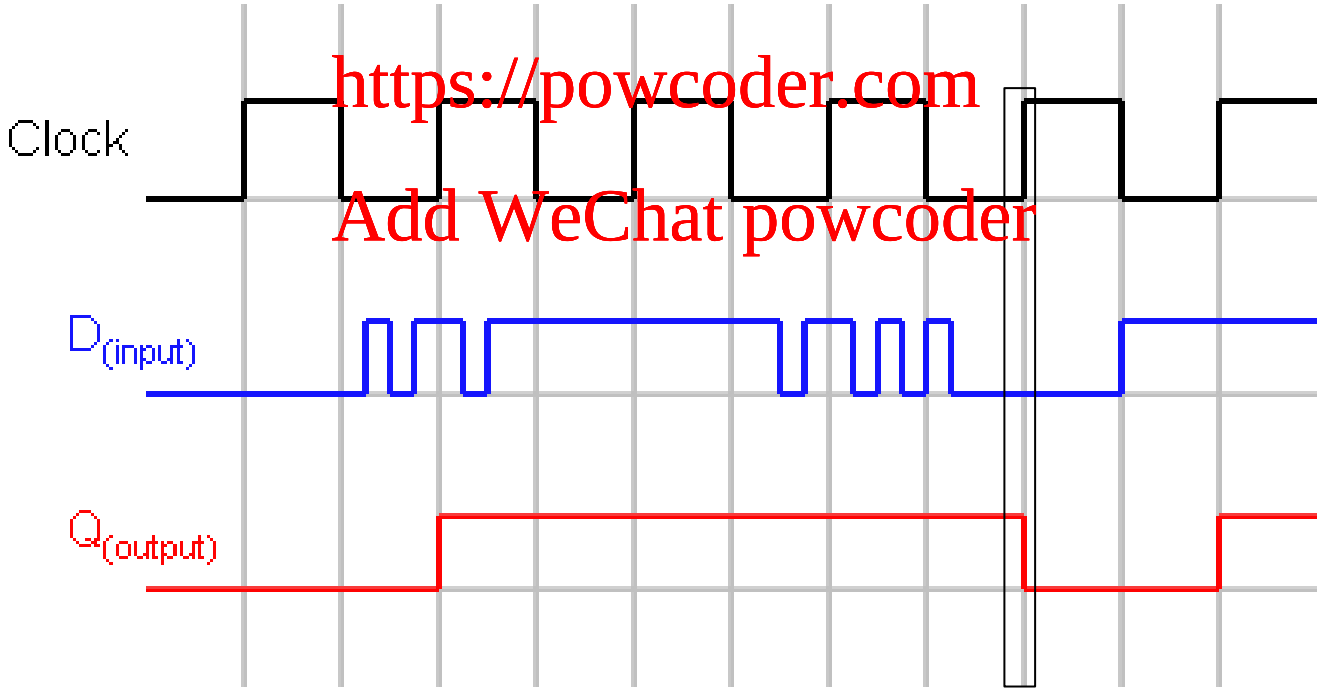
D flip-flop(positive-edge) timing diagram

- Q becomes D at positive clk edge.
  - Stores value until next positive clk edge.
- clk oscillates between 0 and 1
  - frequency = 1/period



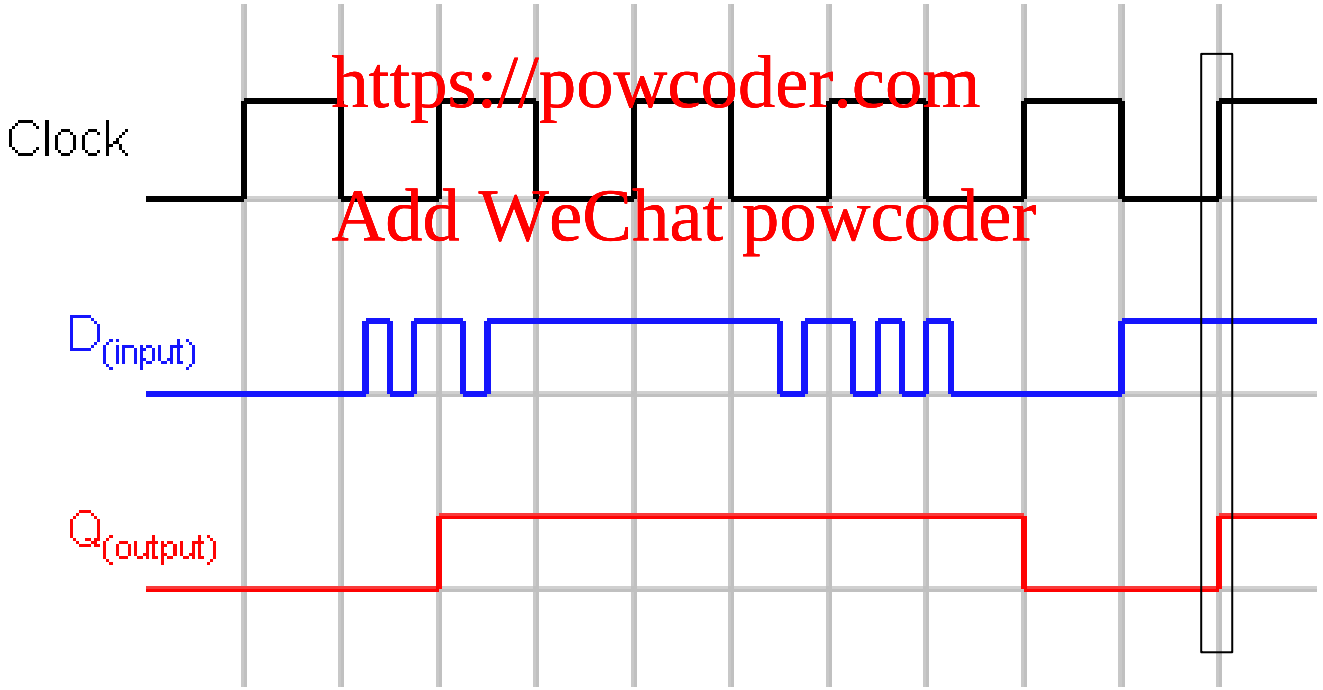
D flip-flop(positive-edge) timing diagram

- Q becomes D at positive clk edge.
  - Stores value until next positive clk edge.
- clk oscillates between 0 and 1
  - frequency = 1/period



D flip-flop(positive-edge) timing diagram

- Q becomes D at positive clk edge.
  - Stores value until next positive clk edge.
- clk oscillates between 0 and 1
  - frequency = 1/period

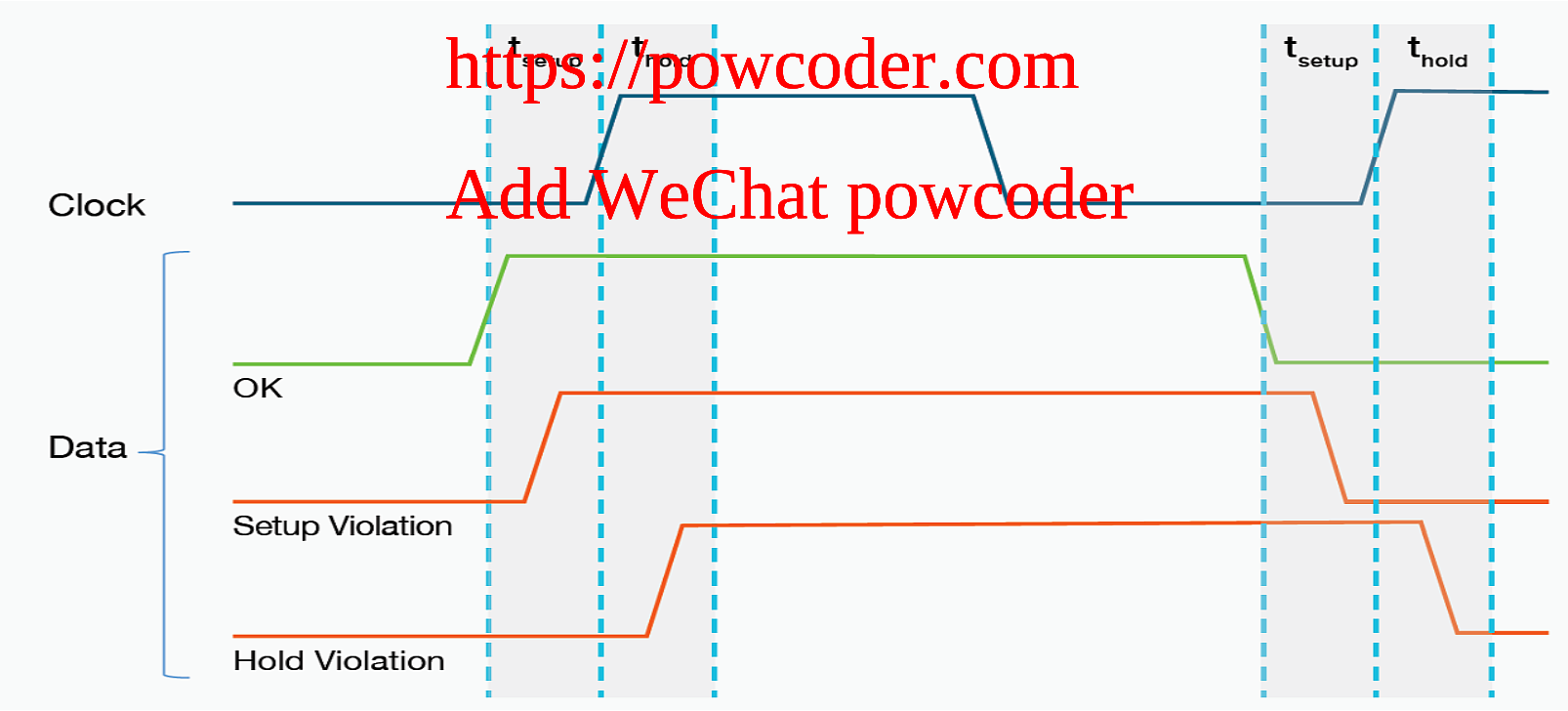


D flip-flop(negative-edge) timing diagram

- Q becomes D at negative clk edge.
  - Stores value until next negative clk edge.
- clk oscillates between 0 and 1
  - frequency = 1/period

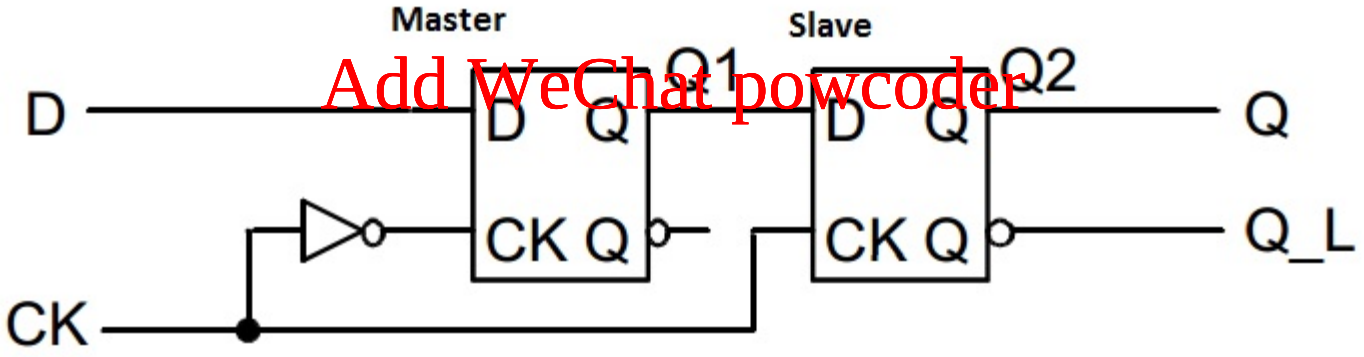
Setup and Hold Time

- Setup time: Time before clock edge where signal has to be stable
- Hold time: Time after clock edge where signal has to be stable



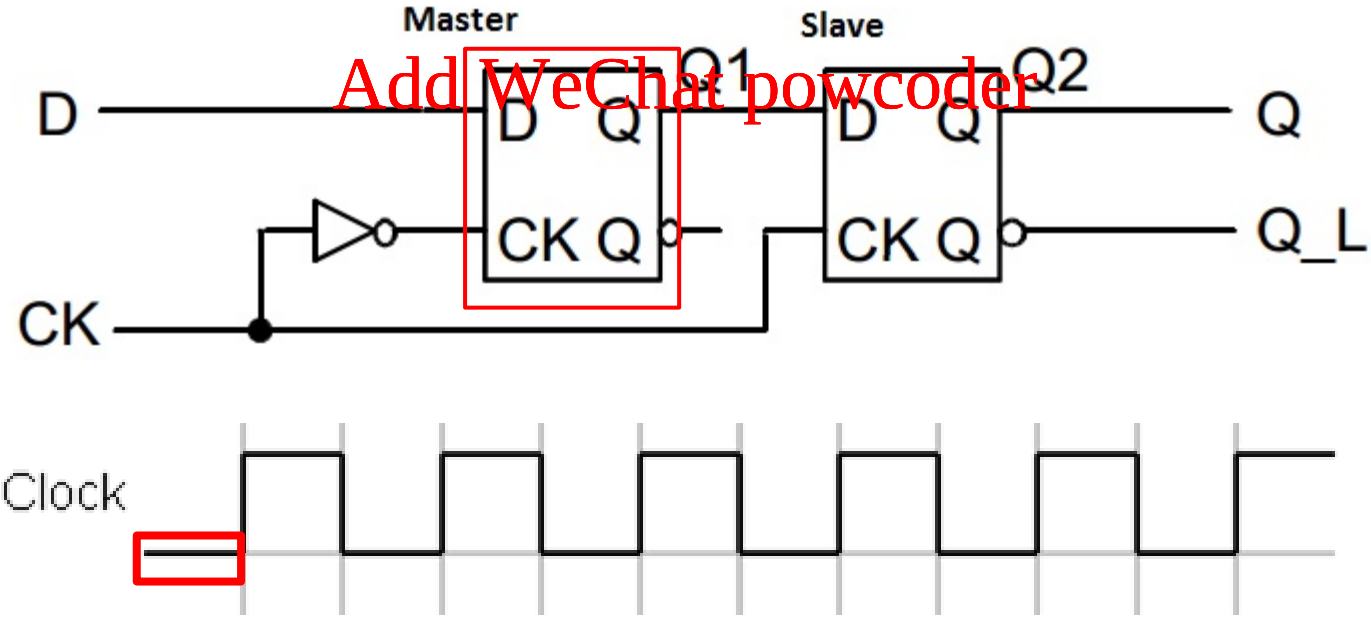
One way to make a Flip-Flop: Two Latches

- One latch(master) is connected to clk' and the other(slave) to clk (positive triggered).
- When clk transitions to high, slave captures last value of master which is held since it's clk is low.



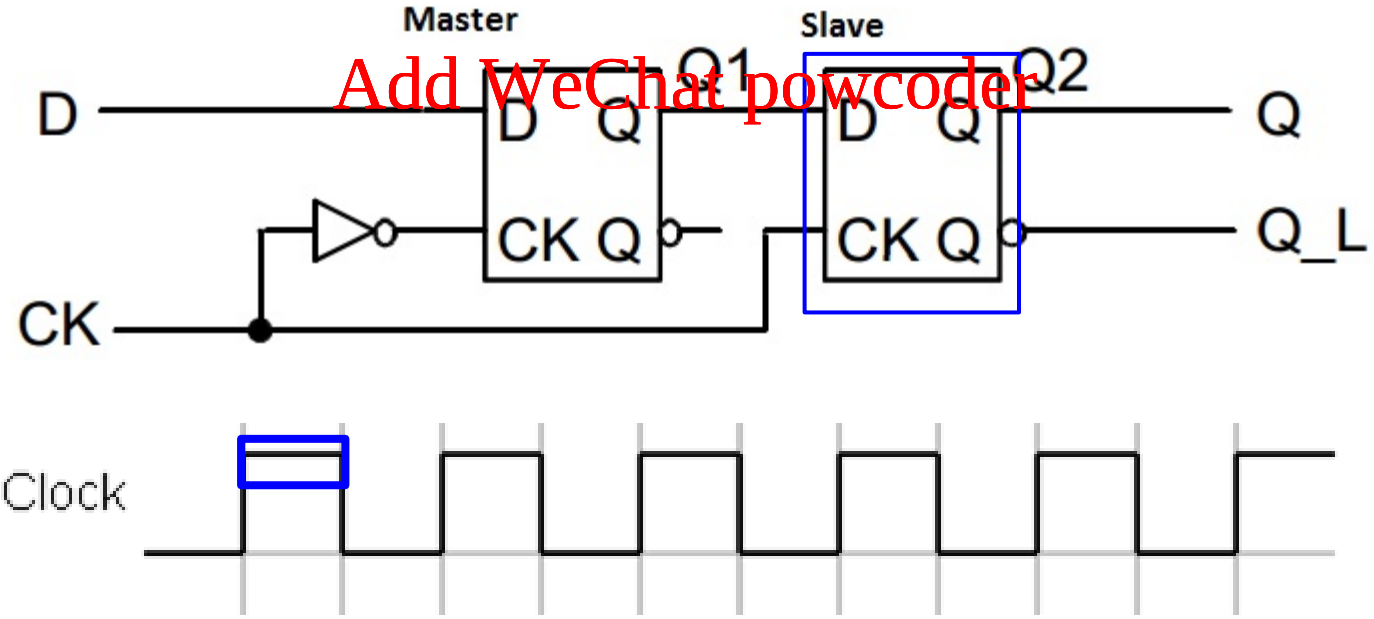
One way to make a Flip-Flop: Two Latches

- One latch(master) is connected to clk' and the other(slave) to clk (positive triggered).
- When clk transitions to high, slave captures last value of master which is now held since it's clk is low.



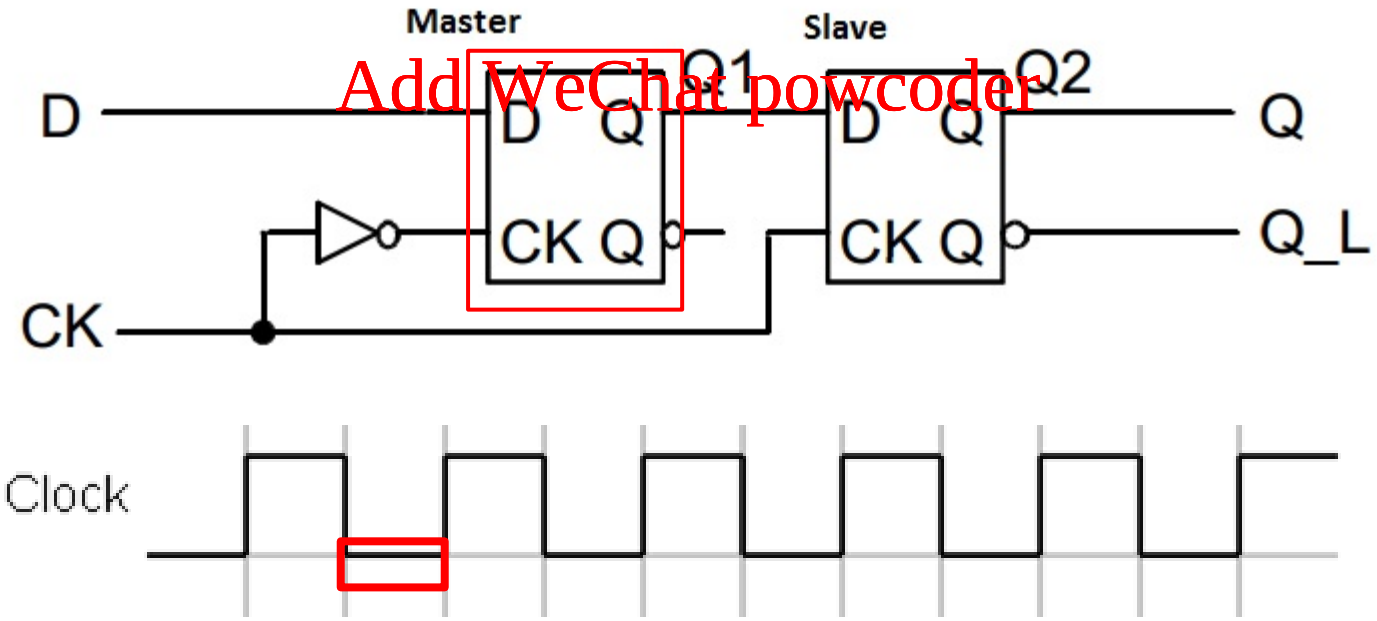
One way to make a Flip-Flop: Two Latches

- One latch(master) is connected to clk' and the other(slave) to clk (positive triggered).
- When clk transitions to high, slave captures last value of master which is held since it's clk is low.



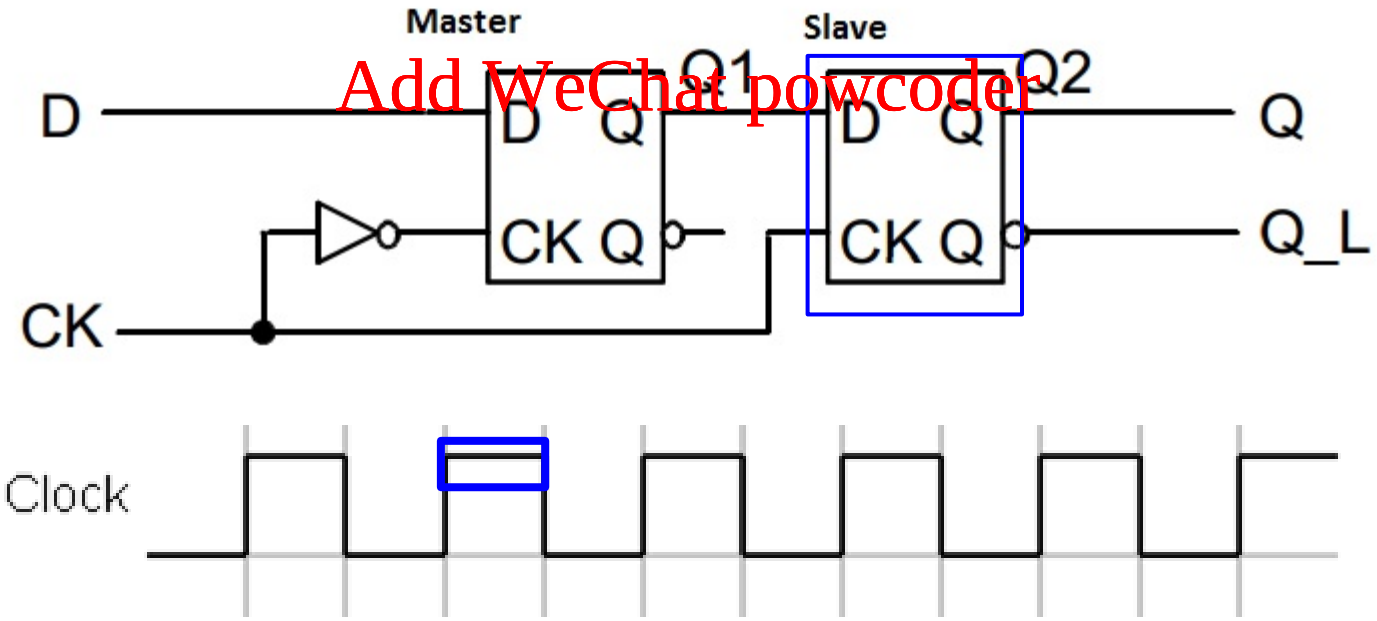
One way to make a Flip-Flop: Two Latches

- When clk transitions to high, slave captures last value of master which is now stored since it's clk is low.
- When clk transitions to low, master is open again but slave is closed, retaining value



One way to make a Flip-Flop: Two Latches

- One latch(master) is connected to clk' and the other(slave) to clk (positive triggered).
- When clk transitions to high, slave captures last value of master which is held since it's clk is low.

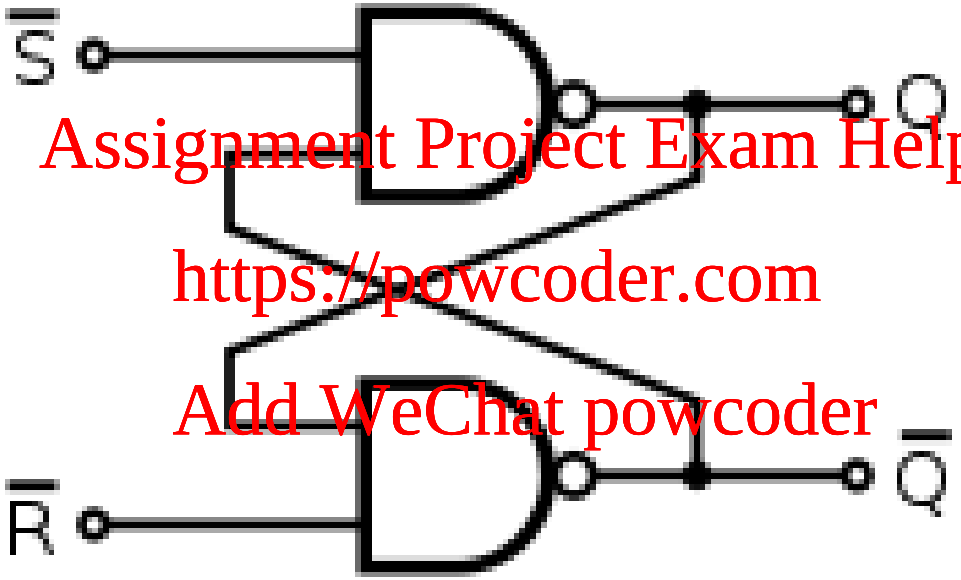


Reset-Set (RS) Latch – or $\overline{SR}$

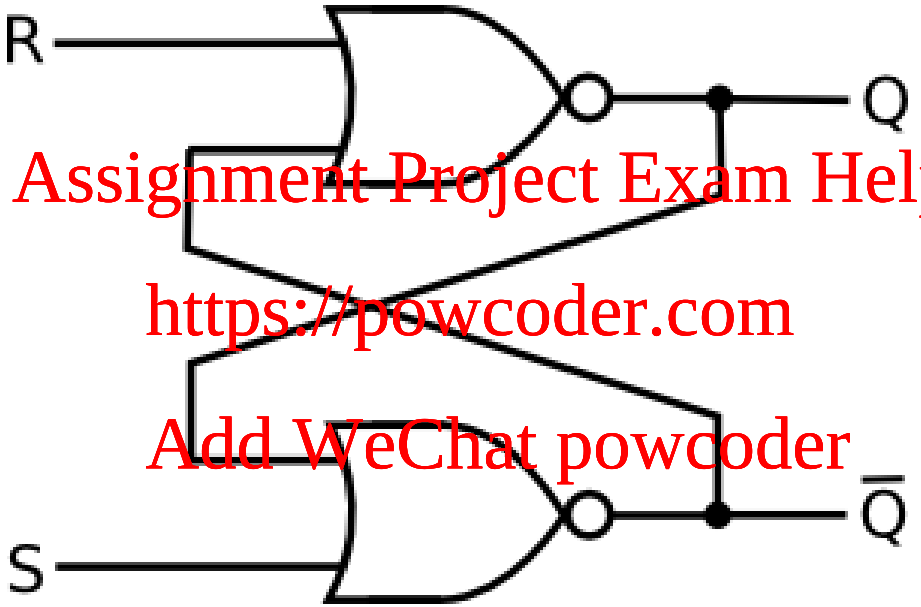
- Two inputs: Set and Reset
- Set to 0 one of the two inputs at a time to store a value, S sets, R clears
- The transition to 00 generates an undefined output

SR latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

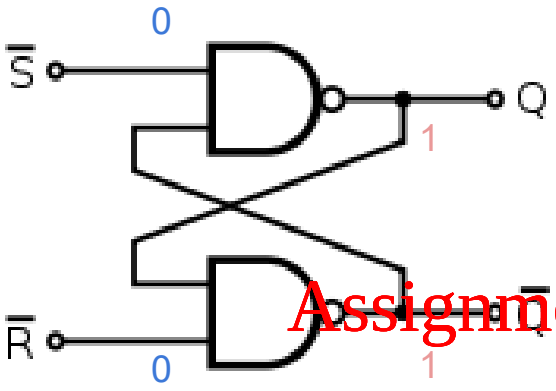
R-S Latch



R-S Latch Nor Gates

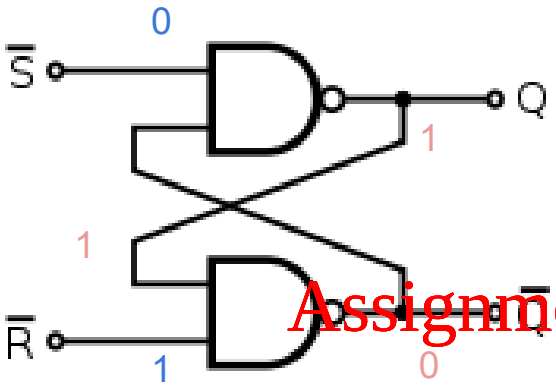


Four SR Latch States: S' 0, R' 0



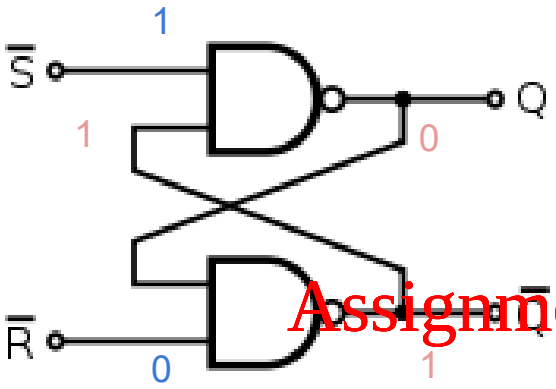
$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

Four SR Latch States: S' 0, R' 1



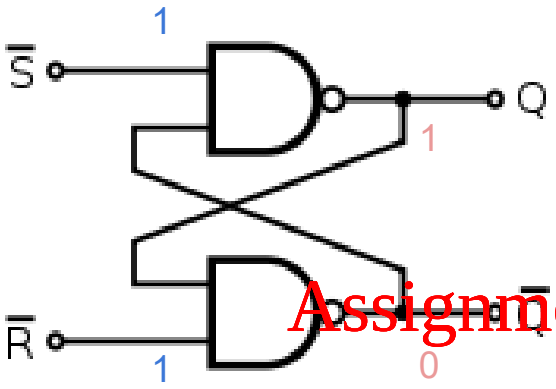
$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

Four SR Latch States: S' 1, R' 0



$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

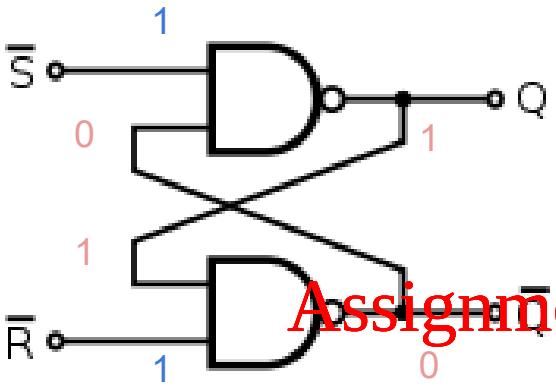
Four SR Latch States: S' 1, R' 1 Memory



$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

Q = 1, Q' = 0

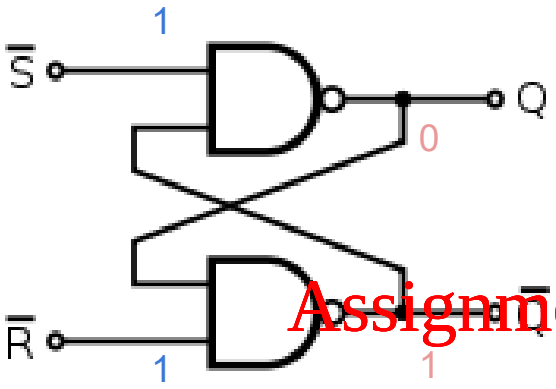
Four SR Latch States: S' 1, R' 1 Memory



$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

Q = 1, Q' = 0

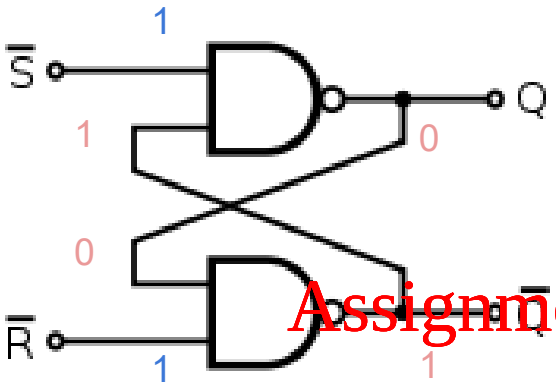
Four SR Latch States: S' 1, R' 1 Memory



$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

Q = 0, Q' = 1

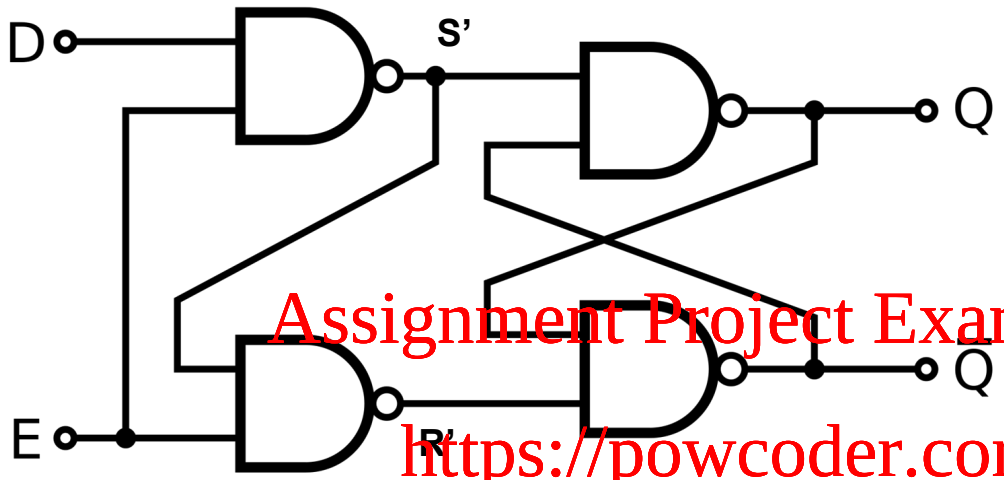
Four SR Latch States: S' 1, R' 1 Memory



$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

Q = 0, Q' = 1

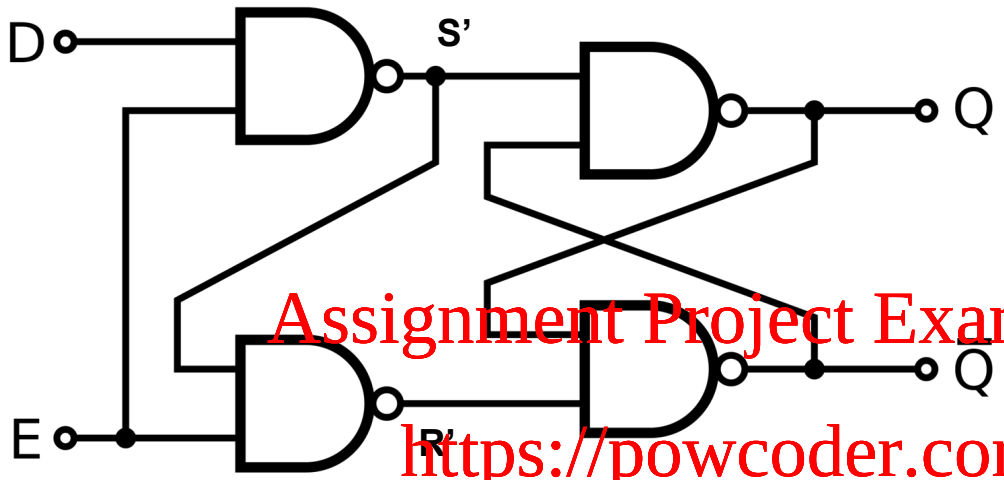
D Latch



$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

E/clk	D	R'	S'	Q	Q'	Comment
0	0	1	1	Q	Q'	Keep state

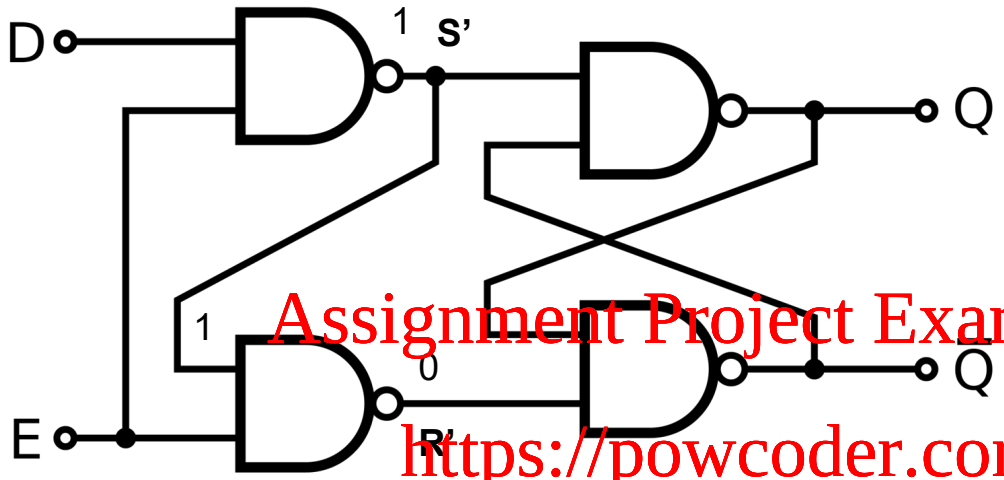
D Latch



$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

E/clk	D	R'	S'	Q	Q'	Comment
0	0	1	1	Q	Q'	Keep state
0	1	1	1	Q	Q'	Keep state

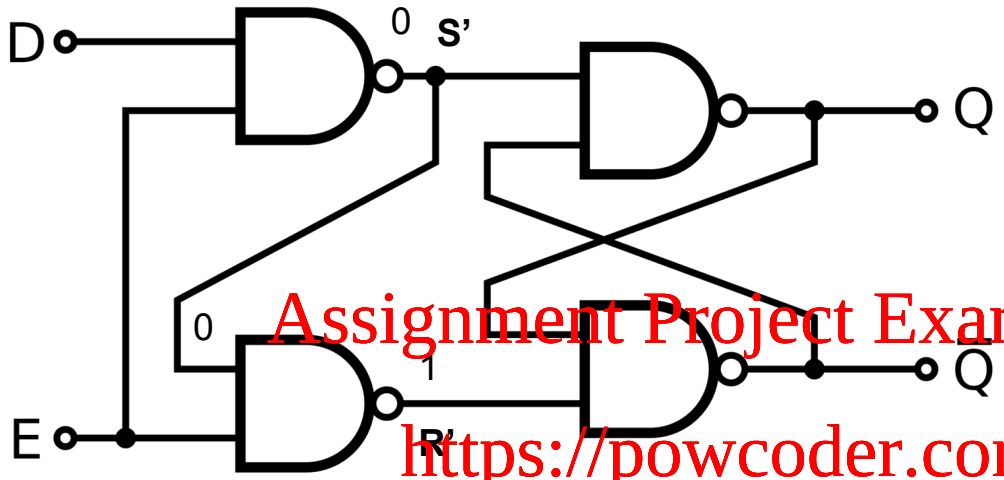
D Latch



$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

E/clk	D	R'	S'	Q	Q'	Comment
0	0	1	1	Q	Q'	Keep state
0	1	1	1	Q	Q'	Keep state
1	0	0	1	0	1	D = Q

D Latch

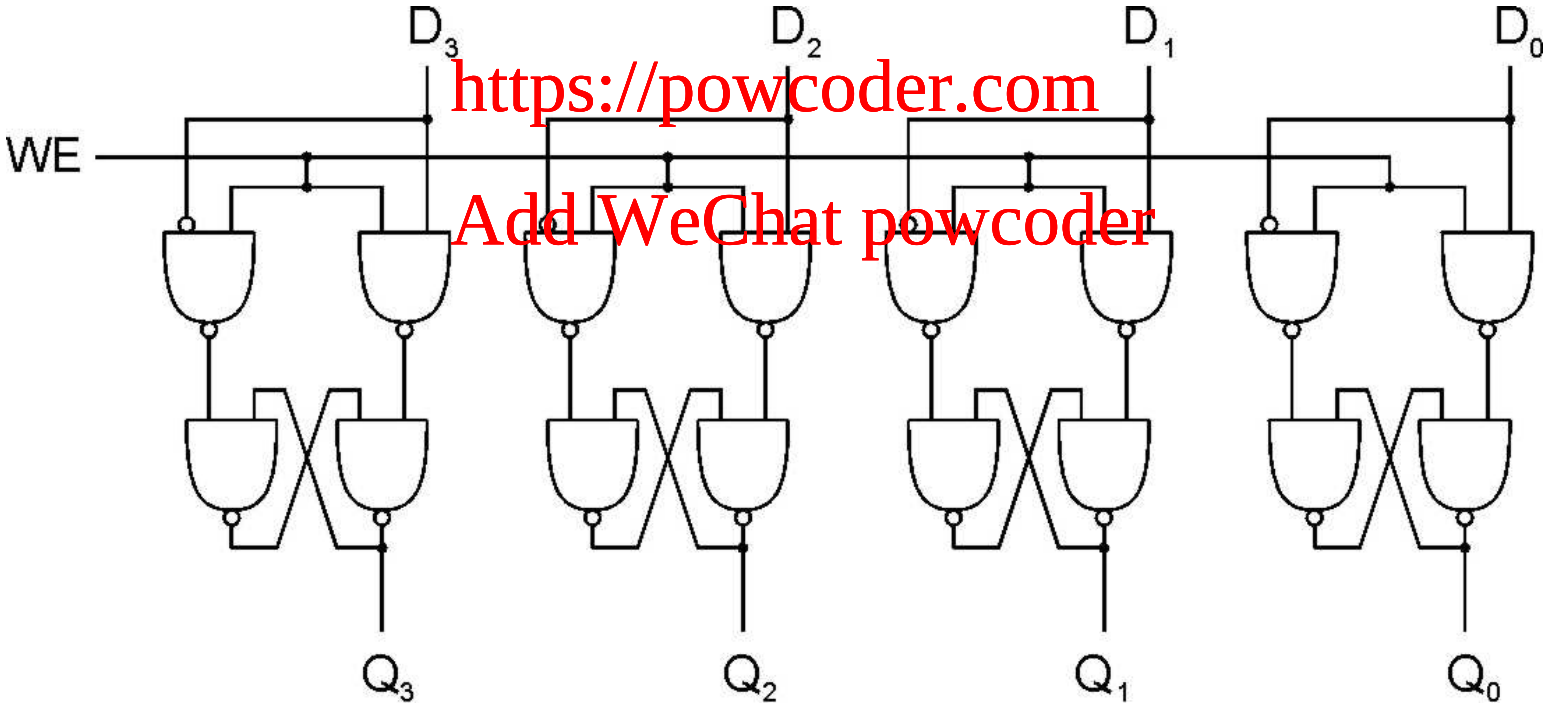


$\overline{SR}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

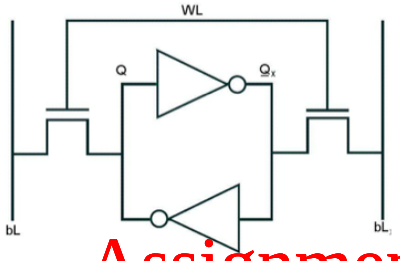
E/clk	D	R'	S'	Q	Q'	Comment
0	0	1	1	Q	Q'	Keep state
0	1	1	1	Q	Q'	Keep state
1	0	0	1	0	1	D = Q
1	1	1	0	1	0	D = Q

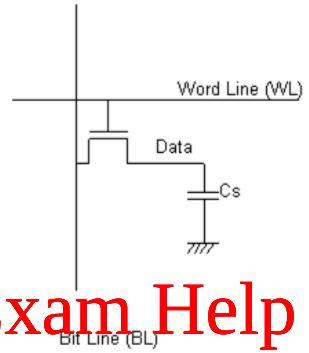
Register

- A register stores a multi-bit value
- Common WE which latches the n-bit value



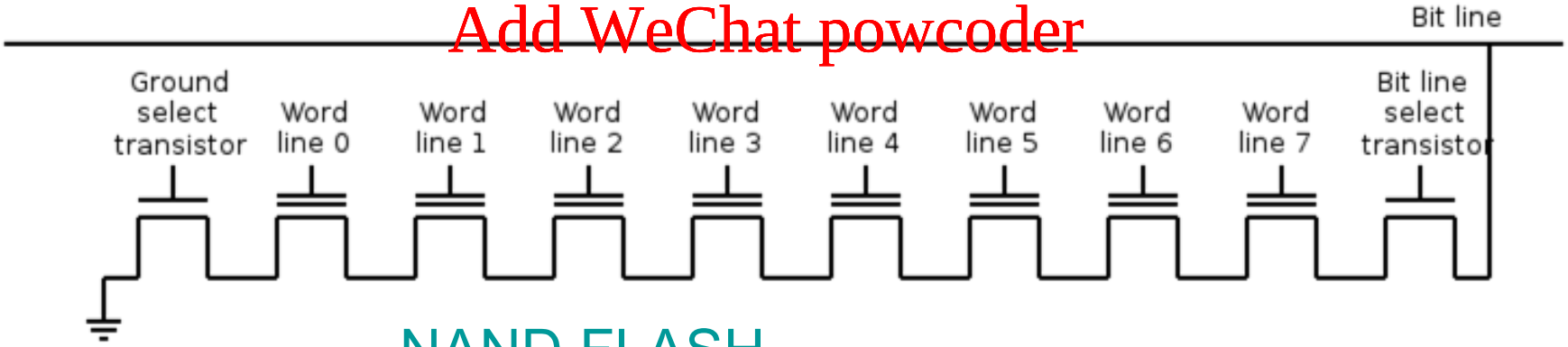
Other types of memory…





SRAM
(on-chip usually)

DRAM
(off-chip usually)



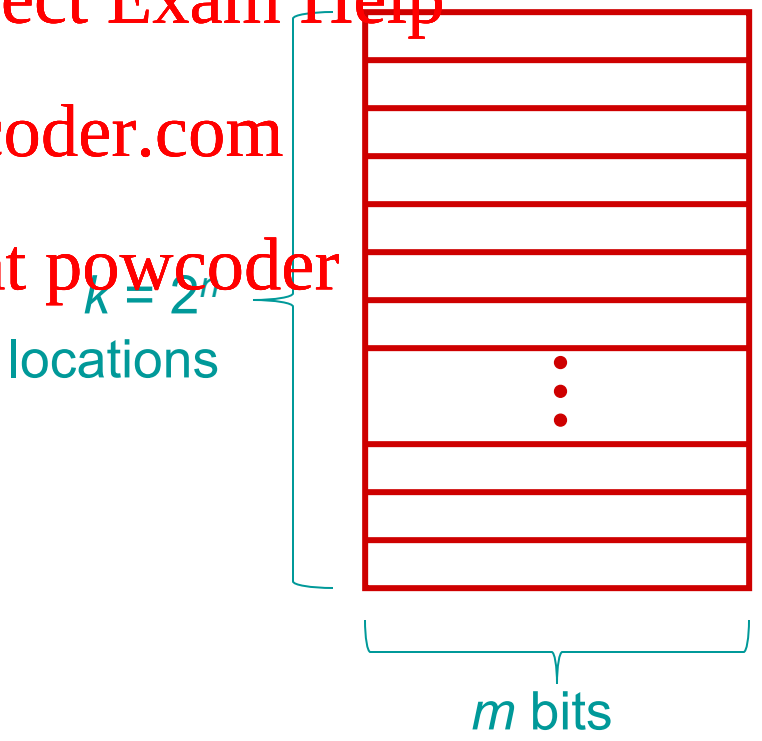
NAND FLASH
(on or off-chip, non-volatile)

Memory

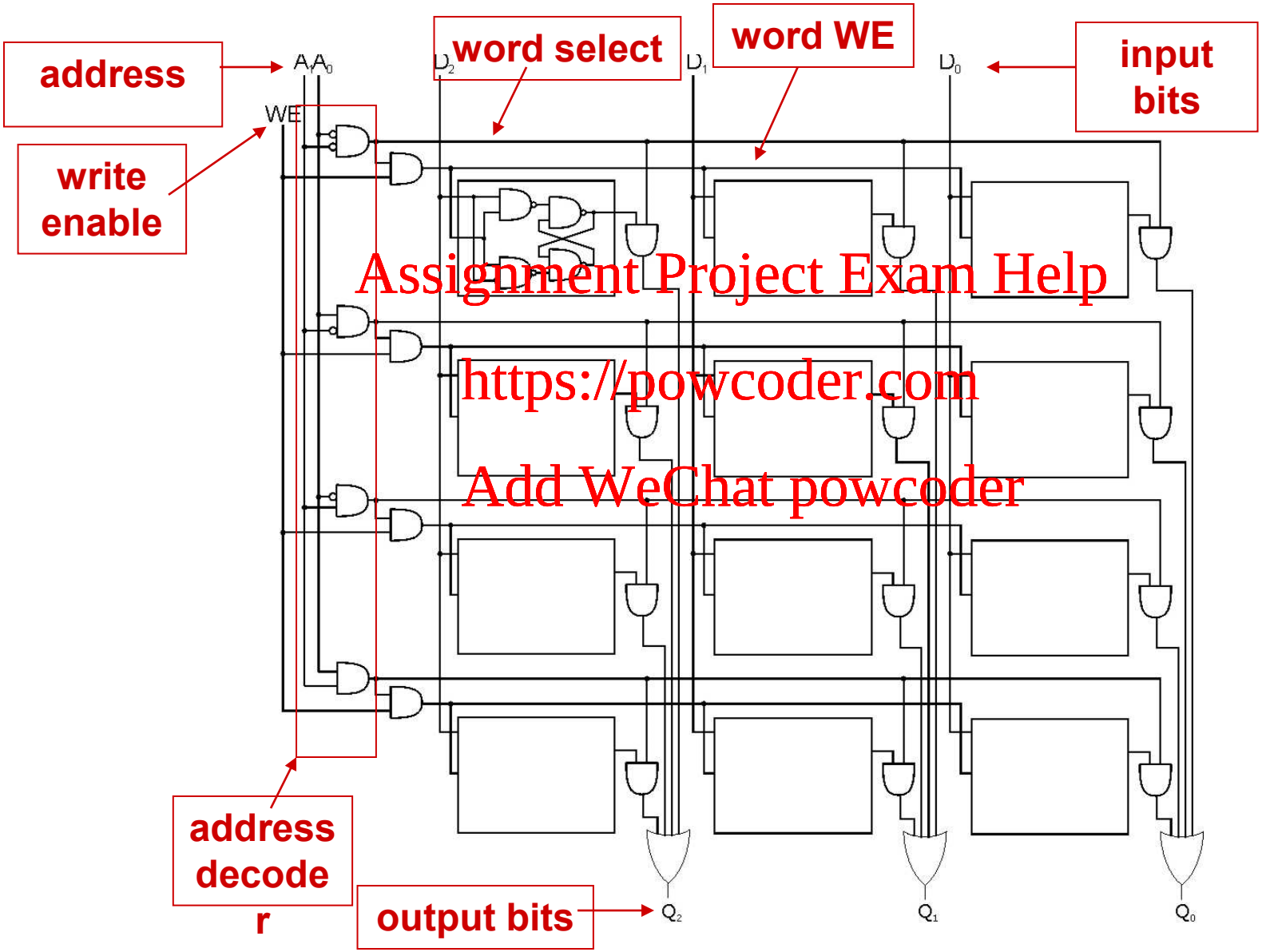
Now that we know how to store bits,
we can build a memory – a logical $k \times m$ array of stored bits.

Address Space:
number of locations
(usually a power of 2)

Addressability:
number of bits per location
(e.g., byte-addressable)

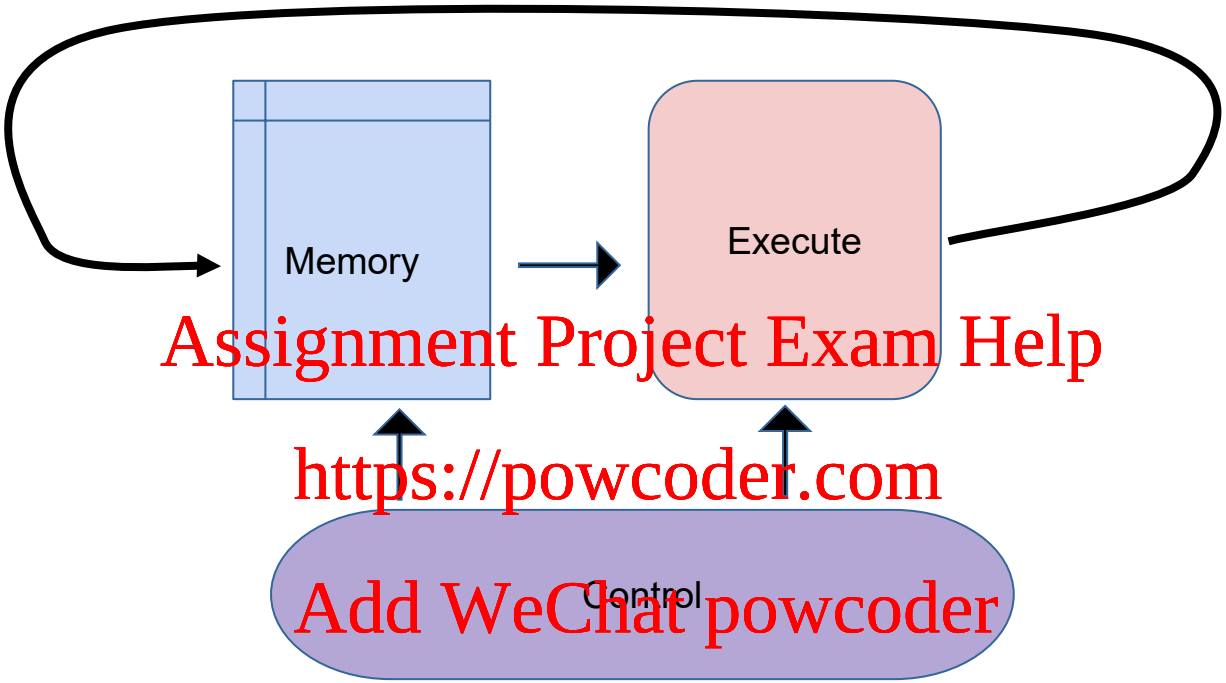


2^2 x 3 Memory



Let's Build a Computer

Basic Computer



Memory: Could be Flip Flops, SRAM/DRAM, Flash etc
Execute: Combinational Logic (Adder, Shifter, Rotation etc)
Control: Finite State Machine (combination of sequential and combinational logic circuits)

Data Path

Combinational Logic

Storage

State Machine