Computer Architecture

Lecture 05

Finite State Machines

Synchronous and Asynchronous Circuits

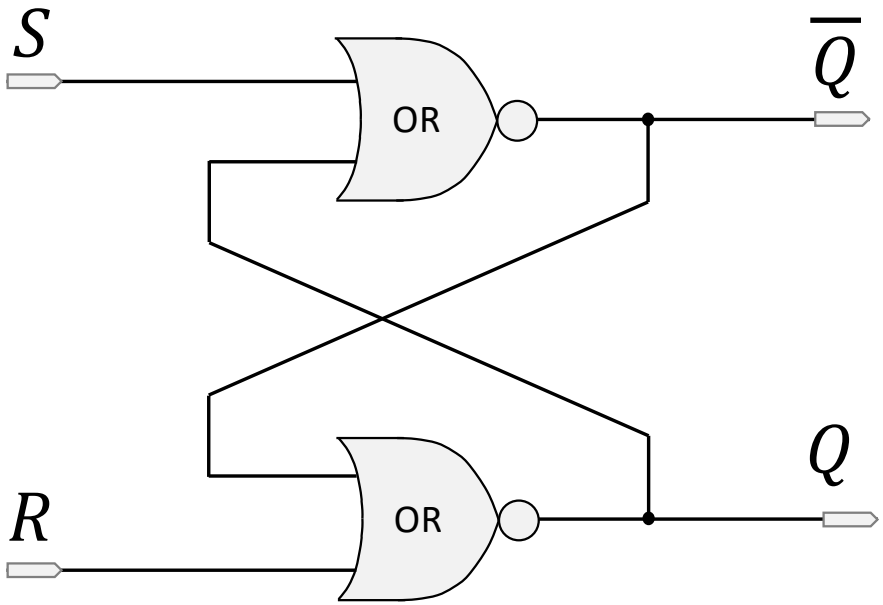
Artem Burmyakov, Alexander Tormasov

September 30, 2021

INNOPOLIS UNIVERSITY

Recap: S/R Latch (or Transparent Latch)

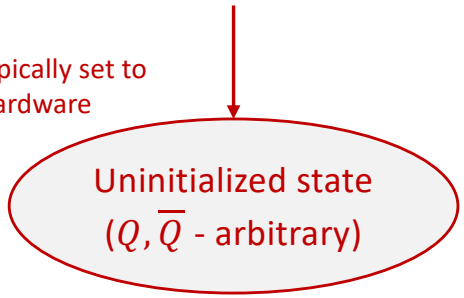
A circuit to store 1 bit of information

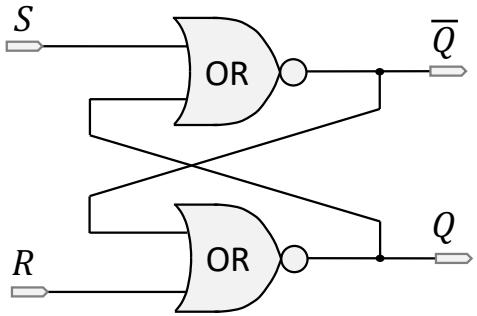


S	***R***	***Q***
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal inputs

State Transition Diagram for a Latch

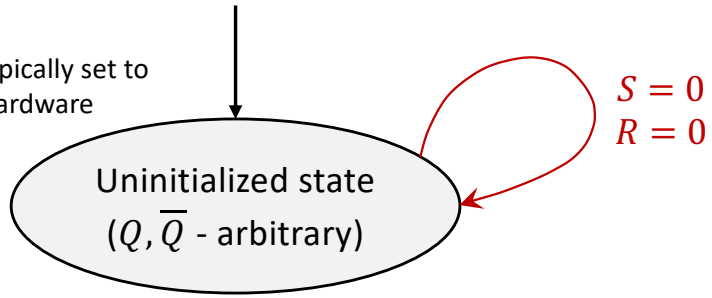
At this state, Q and $\overline{Q}$ are typically set to 0, for most of the availble hardware implementations

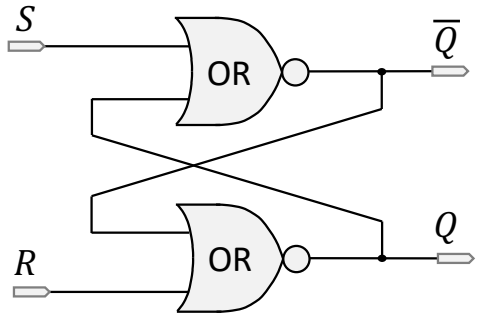




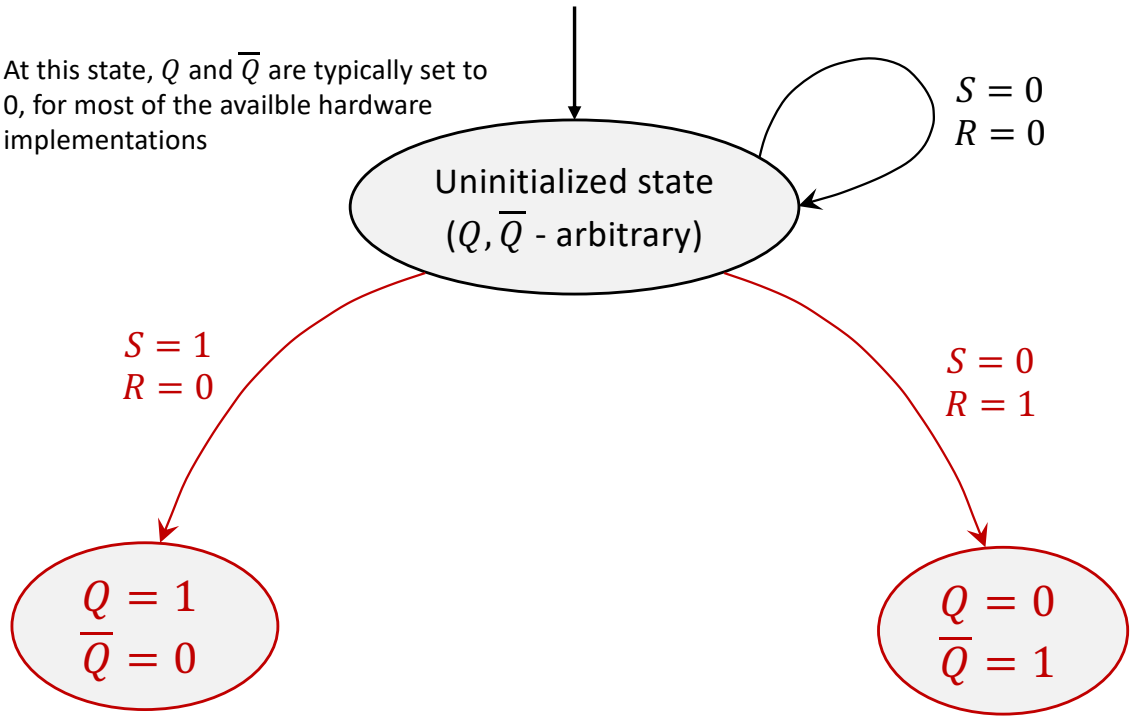
State Transition Diagram for a Latch

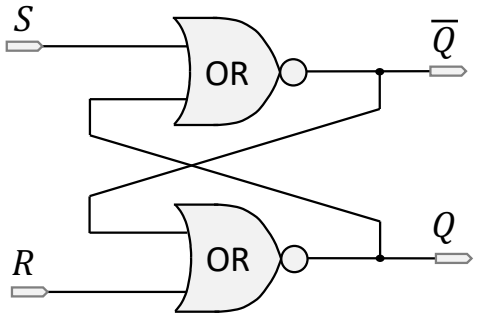
At this state, Q and $\overline{Q}$ are typically set to 0, for most of the availble hardware implementations



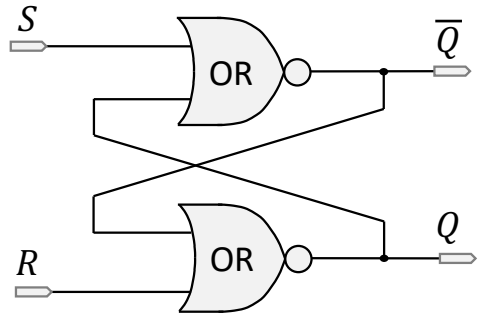


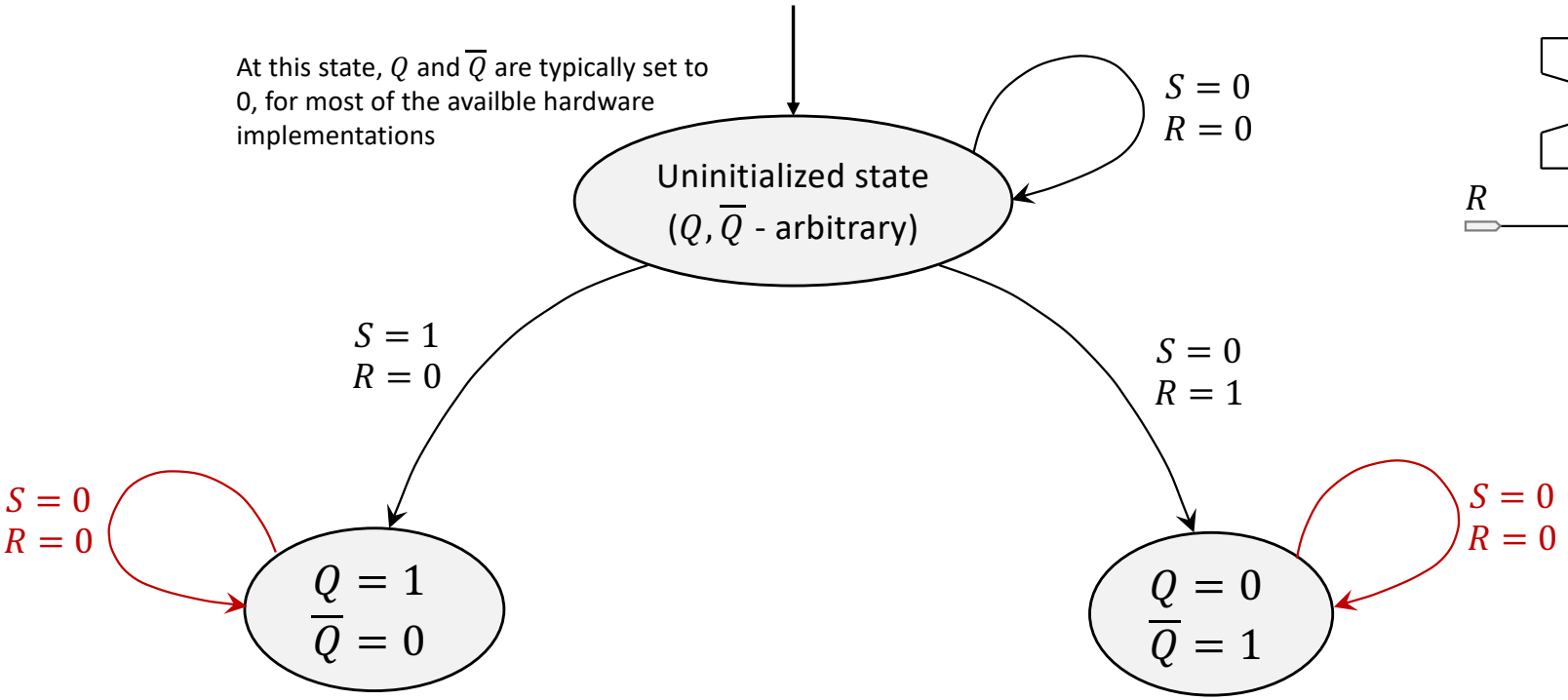
State Transition Diagram for a Latch



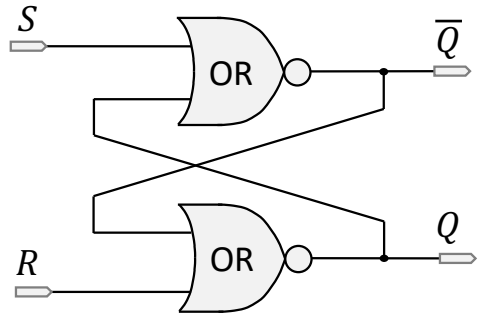


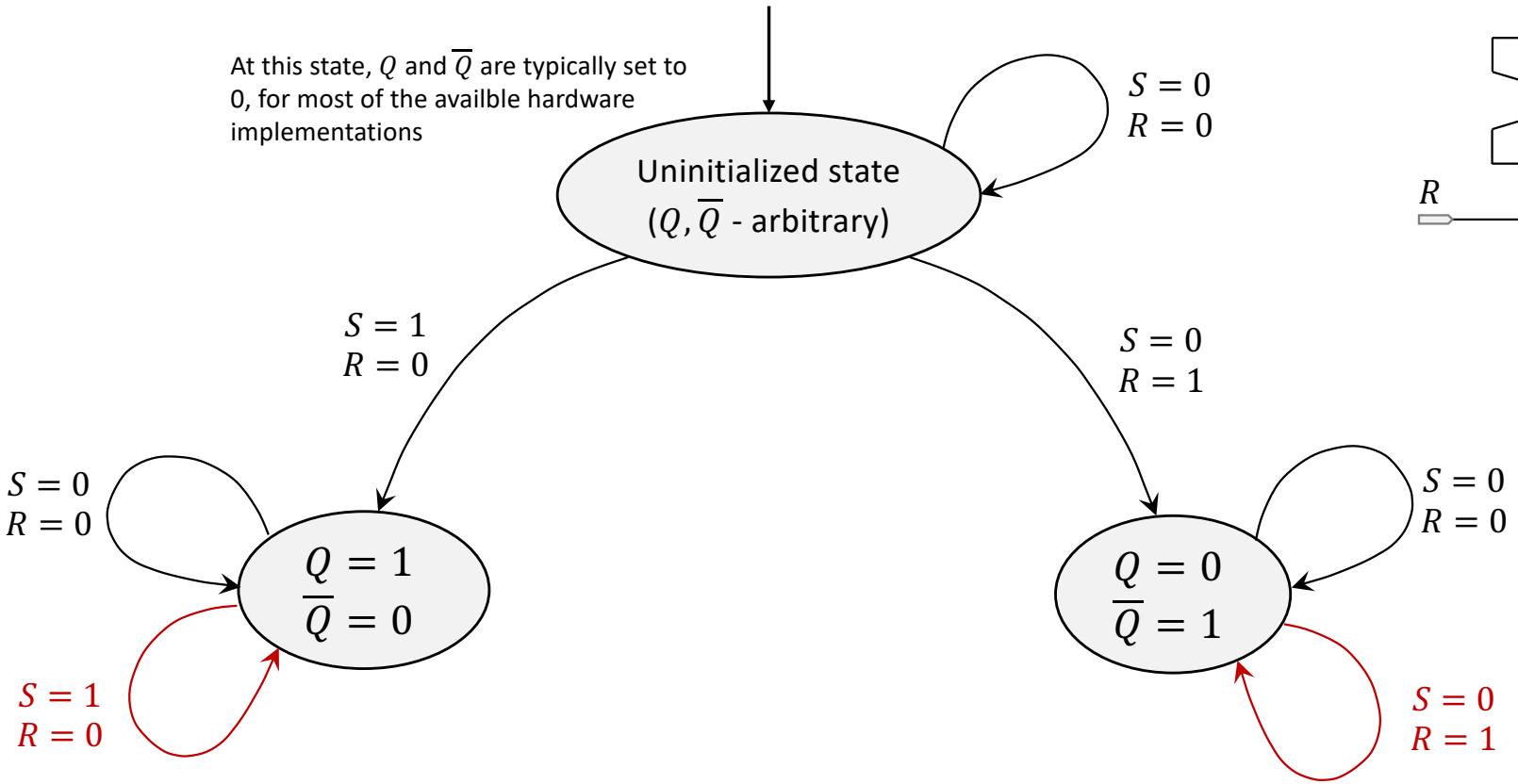
State Transition Diagram for a Latch



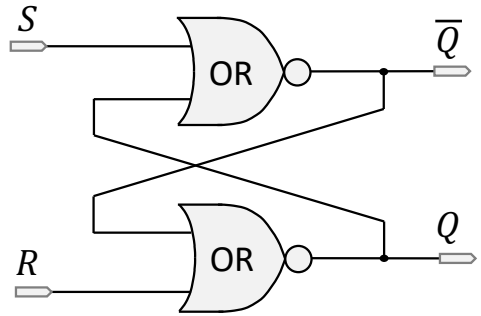


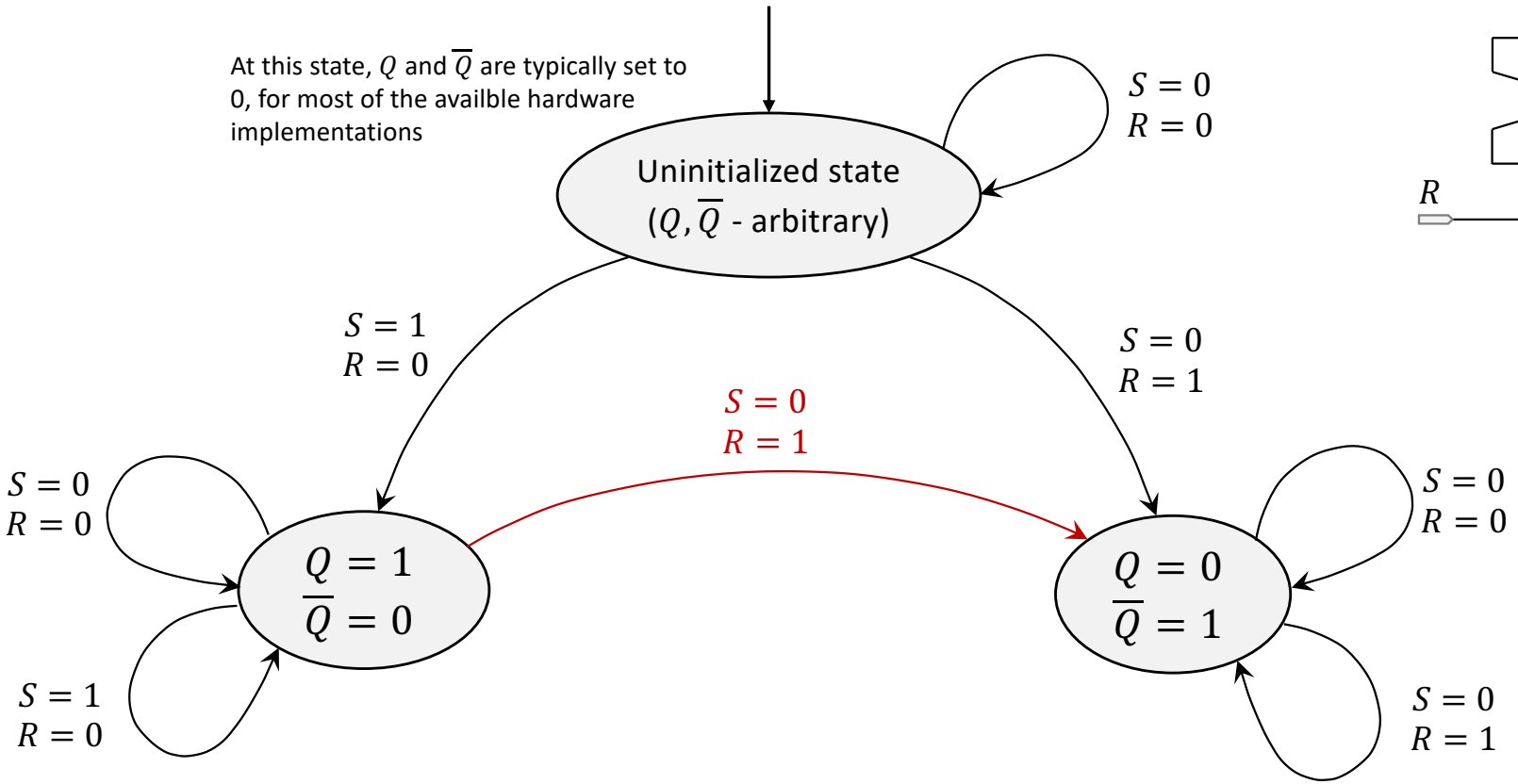
State Transition Diagram for a Latch



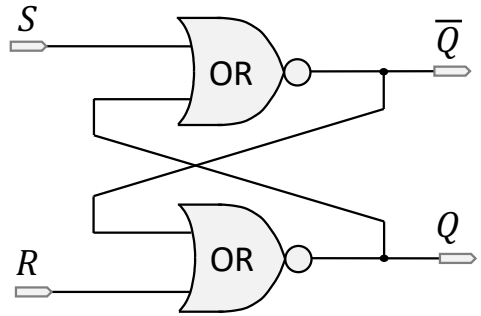


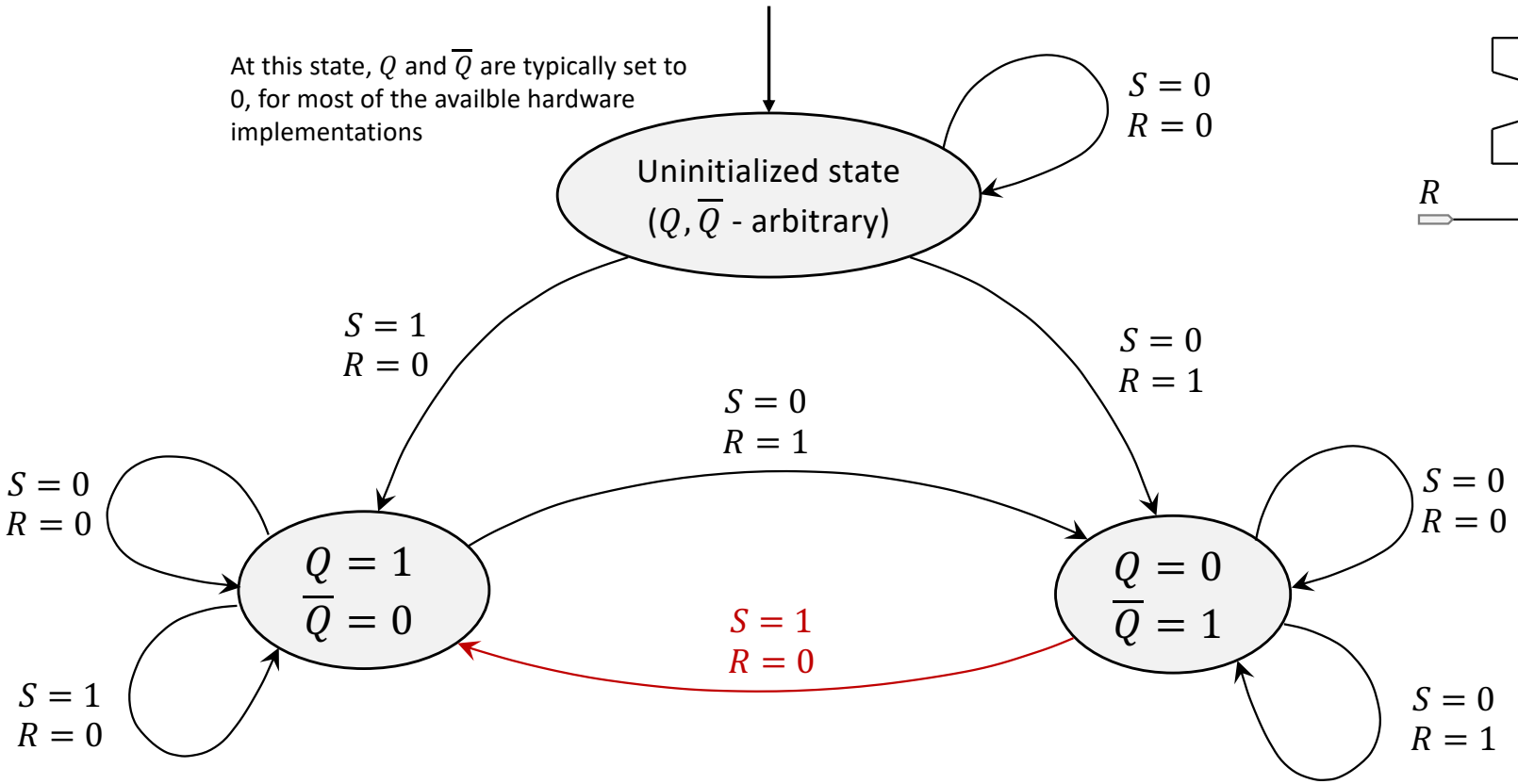
State Transition Diagram for a Latch



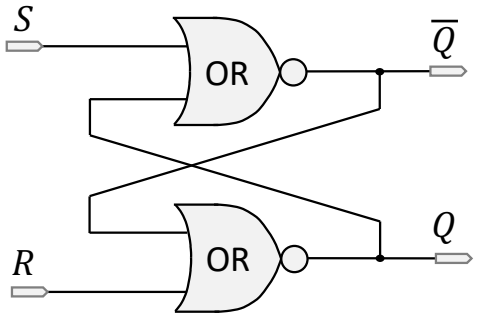


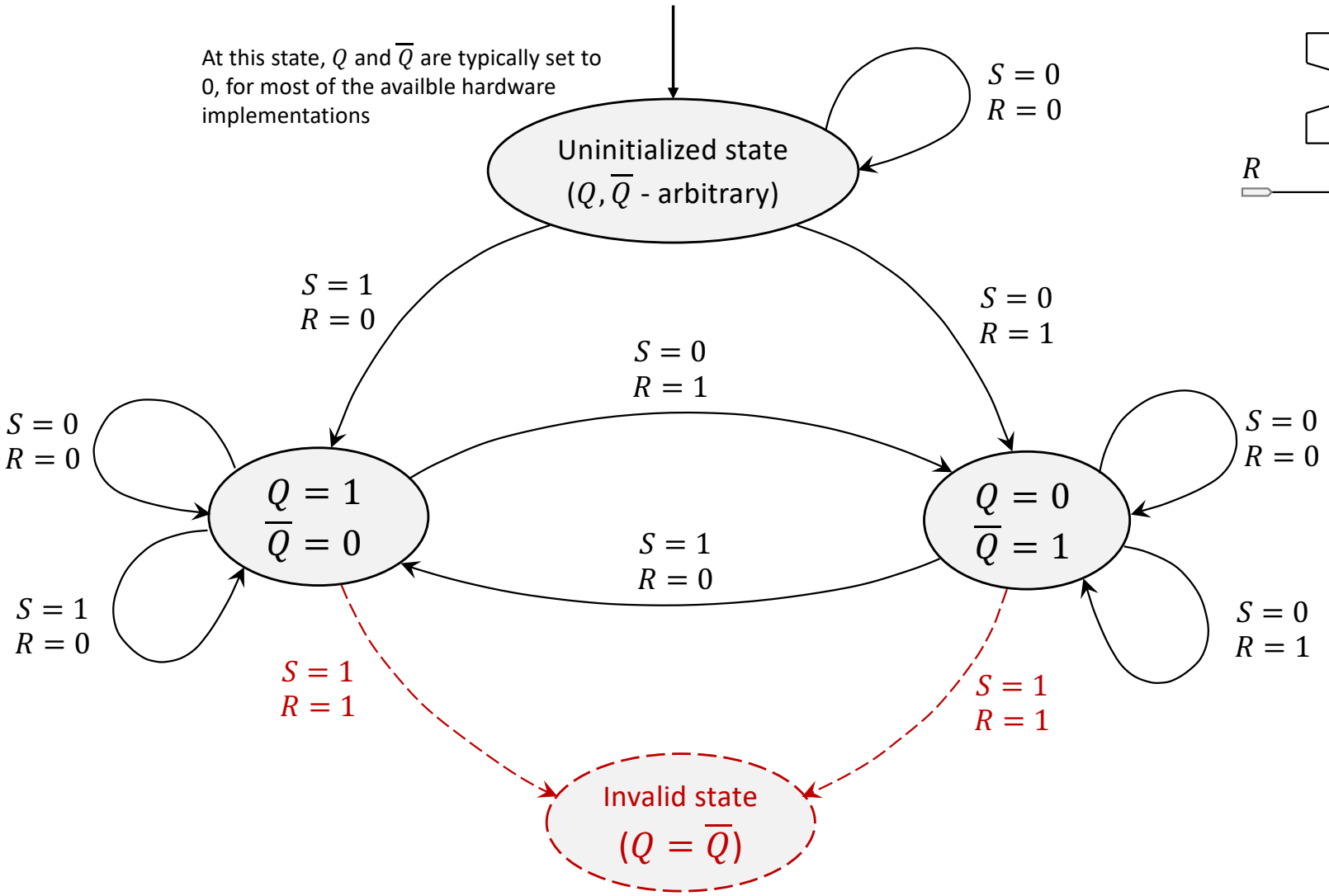
State Transition Diagram for a Latch



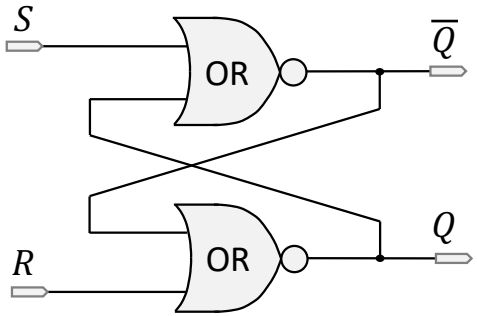


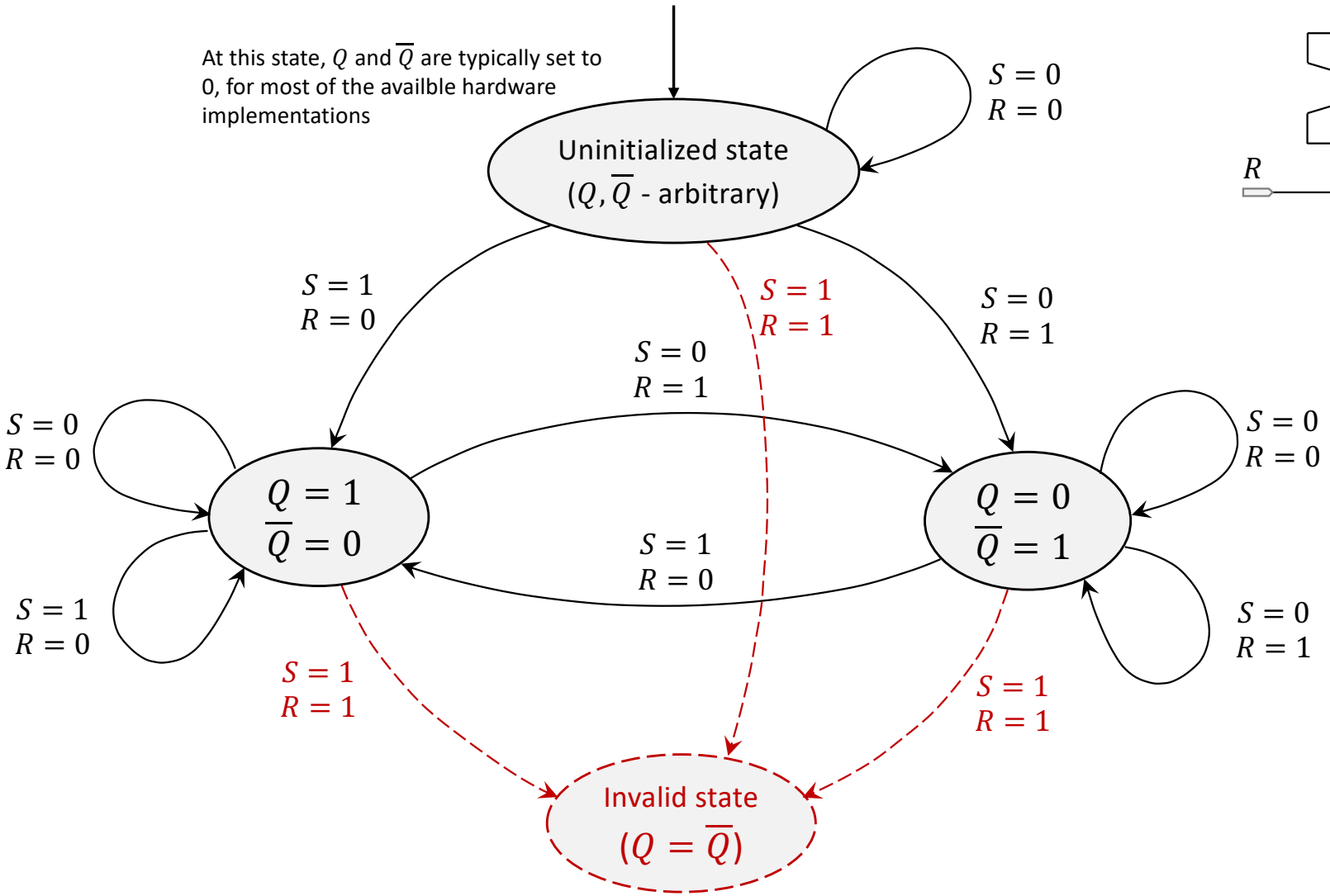
State Transition Diagram for a Latch



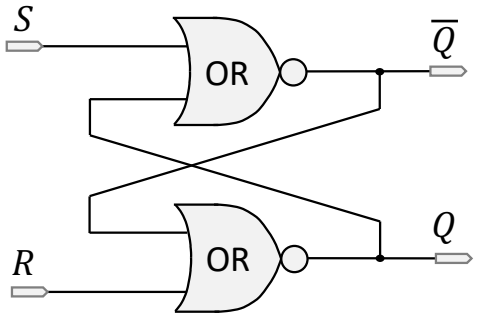


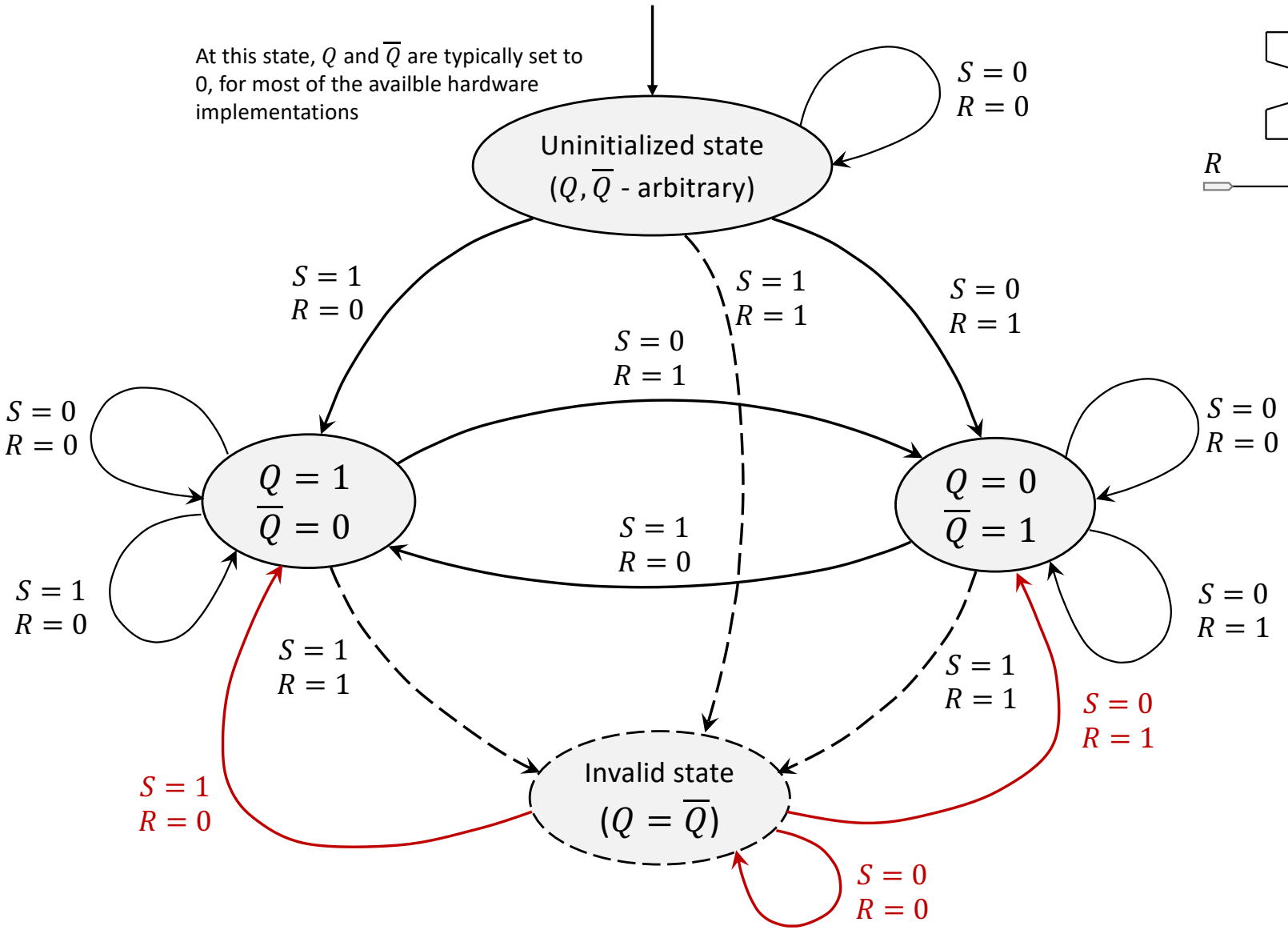
State Transition Diagram for a Latch





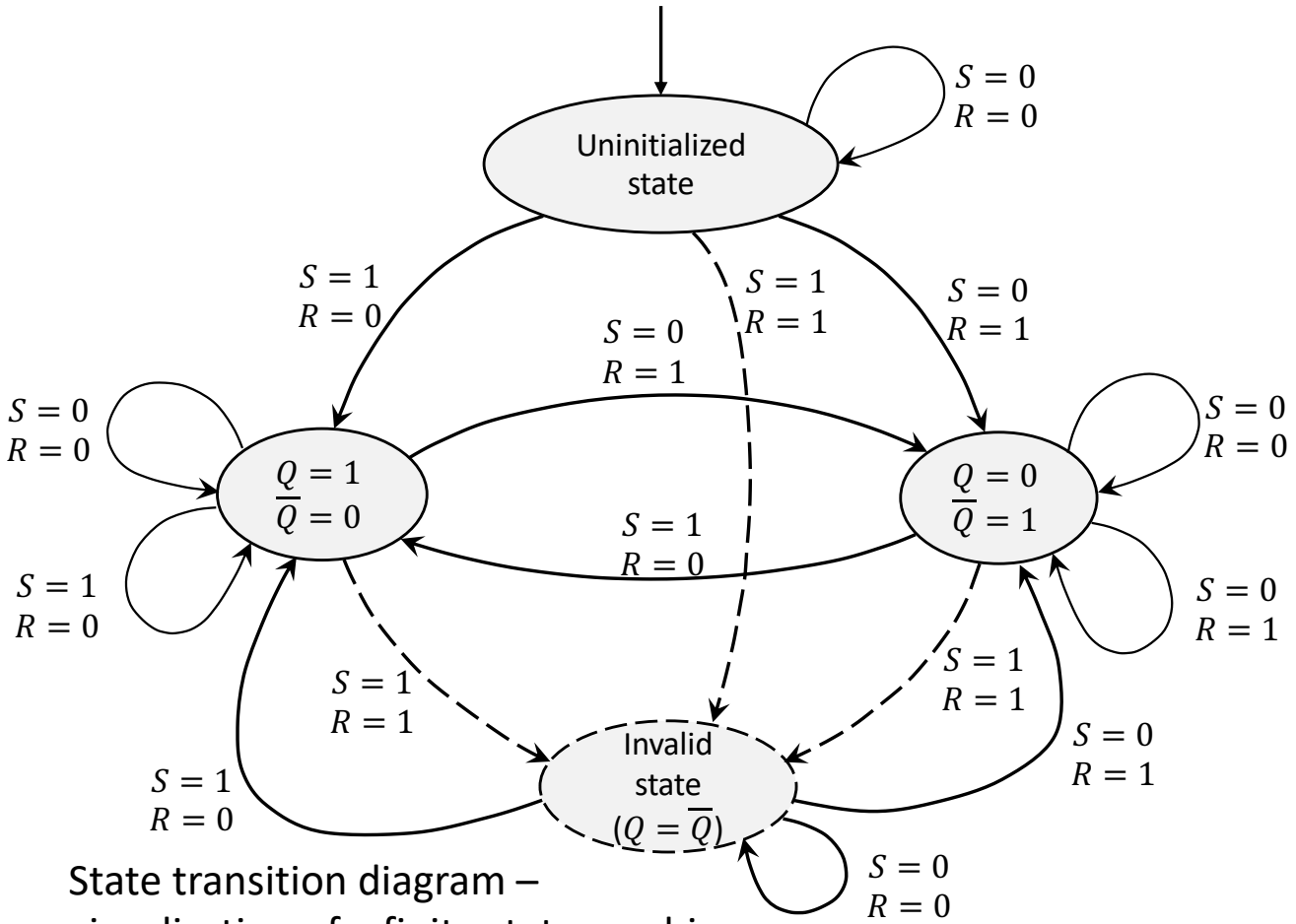
State Transition Diagram for a Latch





Finite State Machine (FSM) (or Finite State Automaton, or simply State Machine)

FSM - an abstract machine

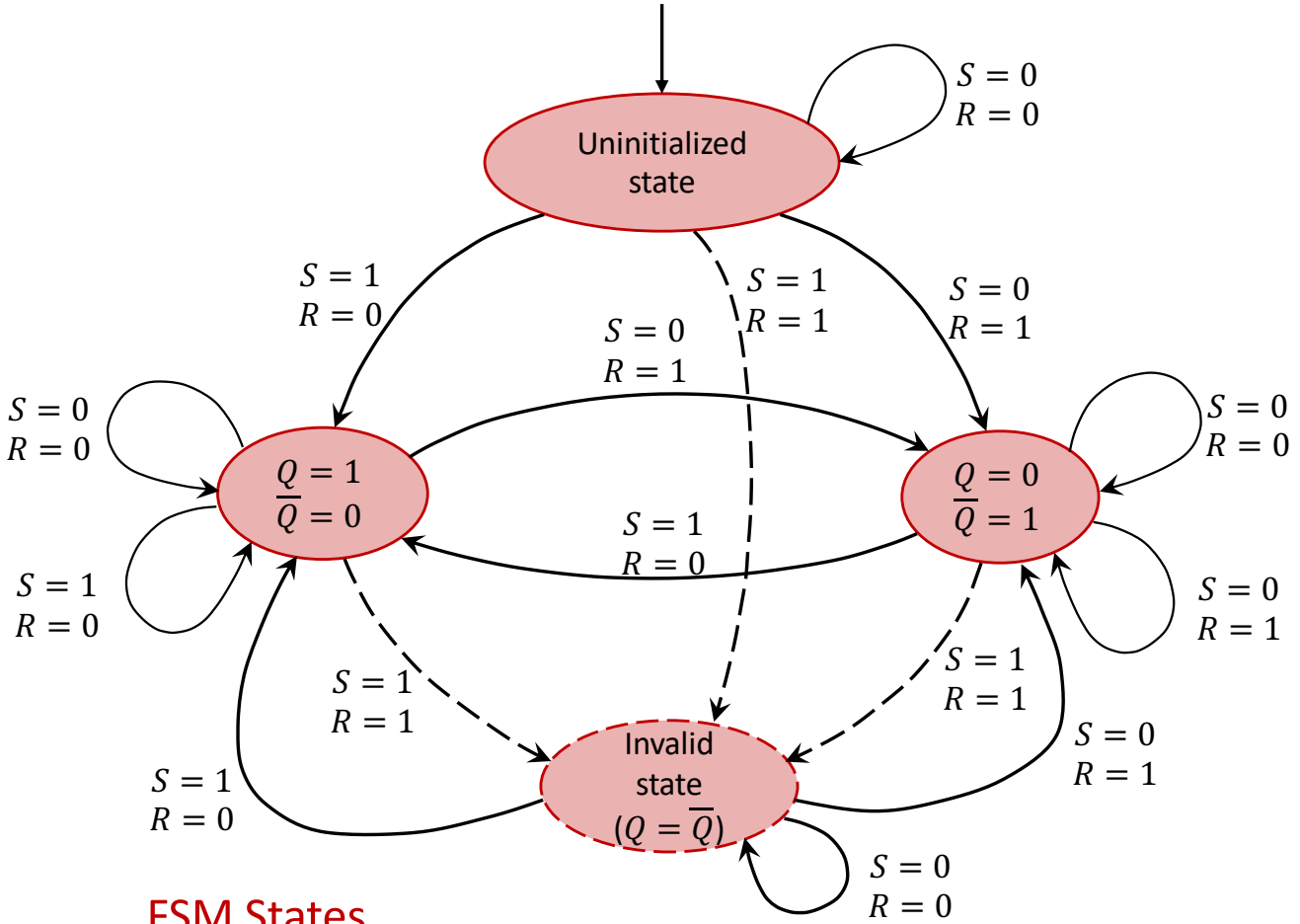


State transition diagram – visualization of a finite state machine

Finite State Machine (FSM) (or Finite State Automaton, or simply State Machine)

FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;



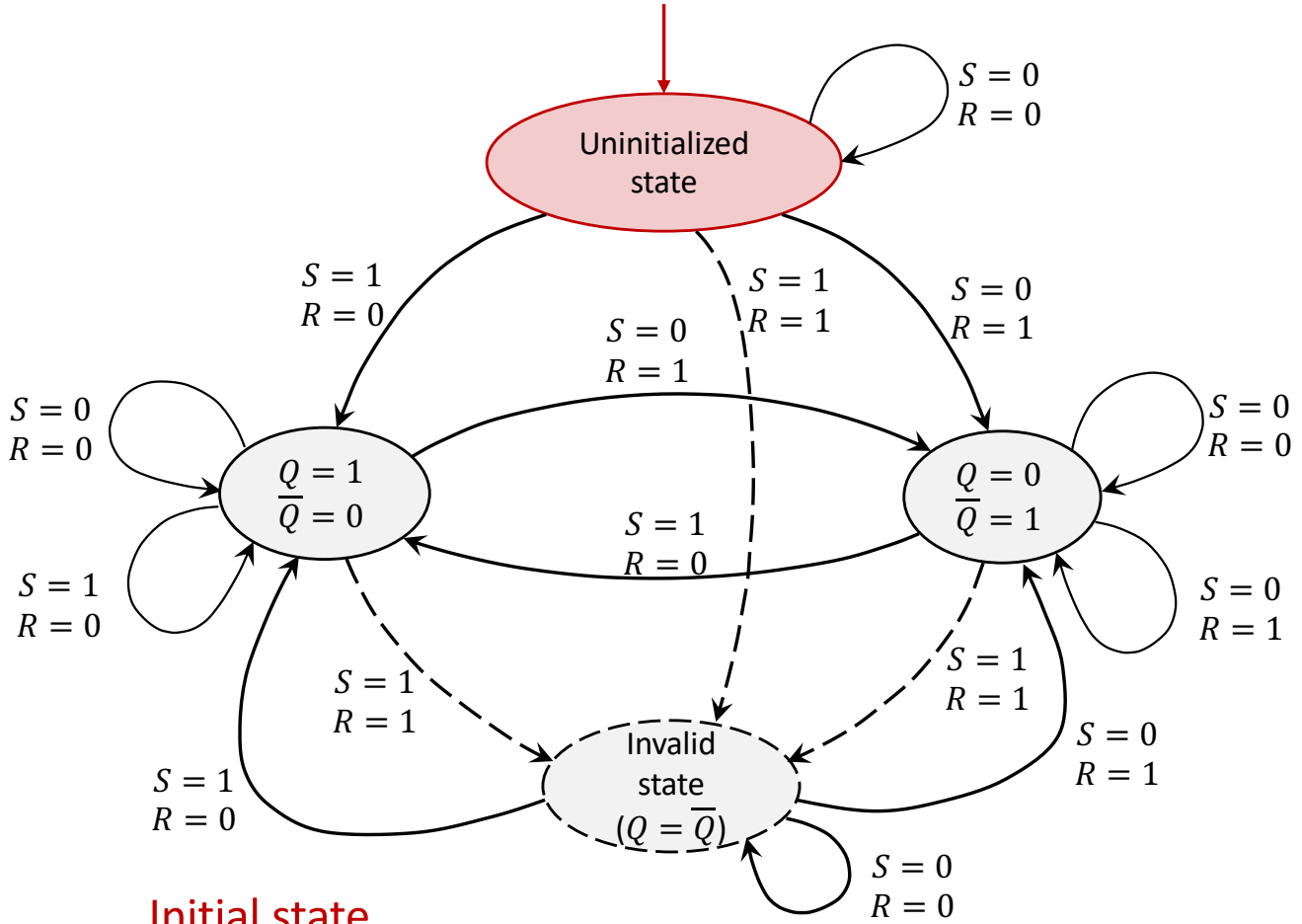
FSM States

Finite State Machine (FSM) (or Finite State Automaton, or simply State Machine)

FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;

b) Has one initial state;



Initial state

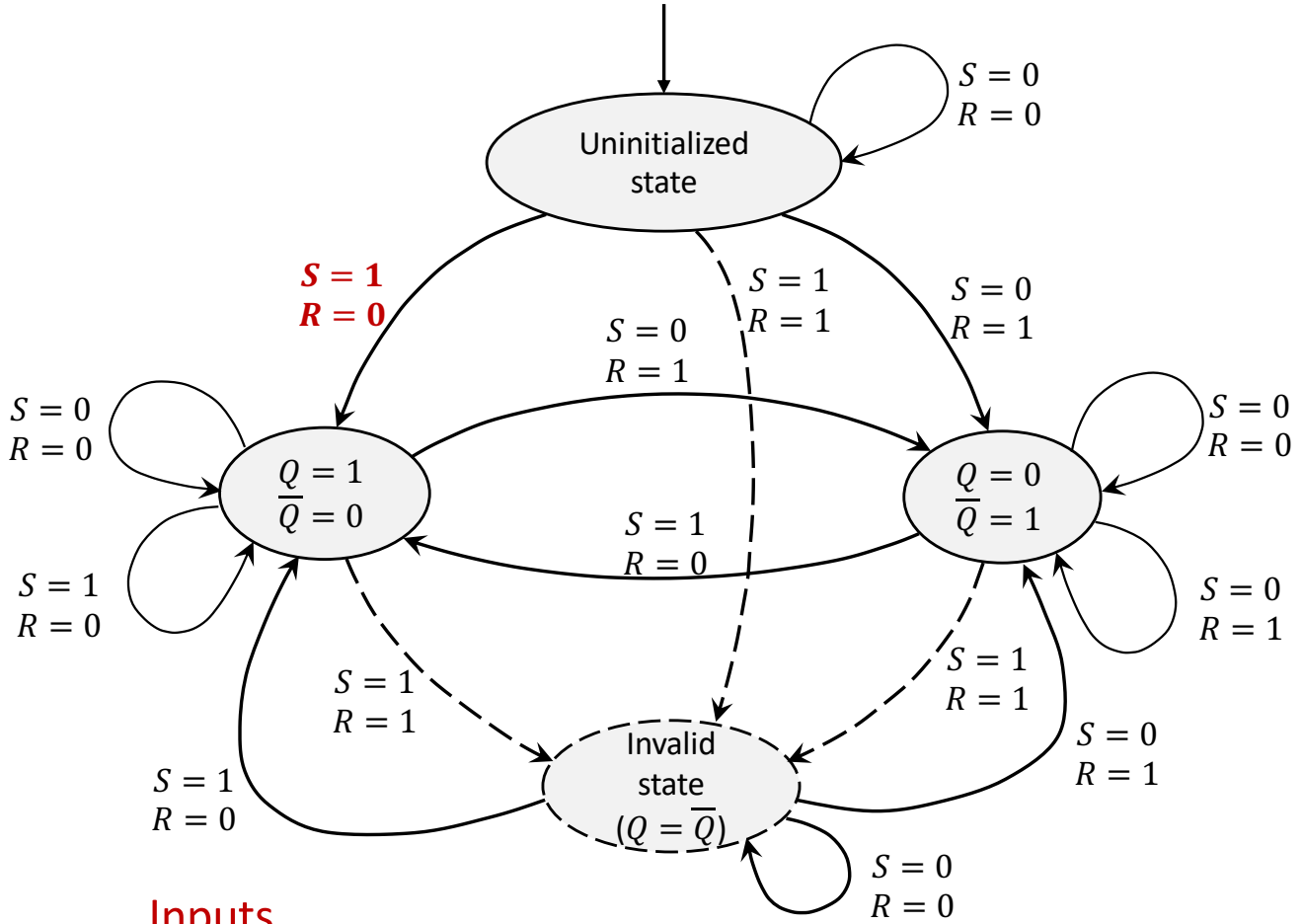
Finite State Machine (FSM) (or Finite State Automaton, or simply State Machine)

FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;

b) Has one initial state;

c) It gets inputs from outside;



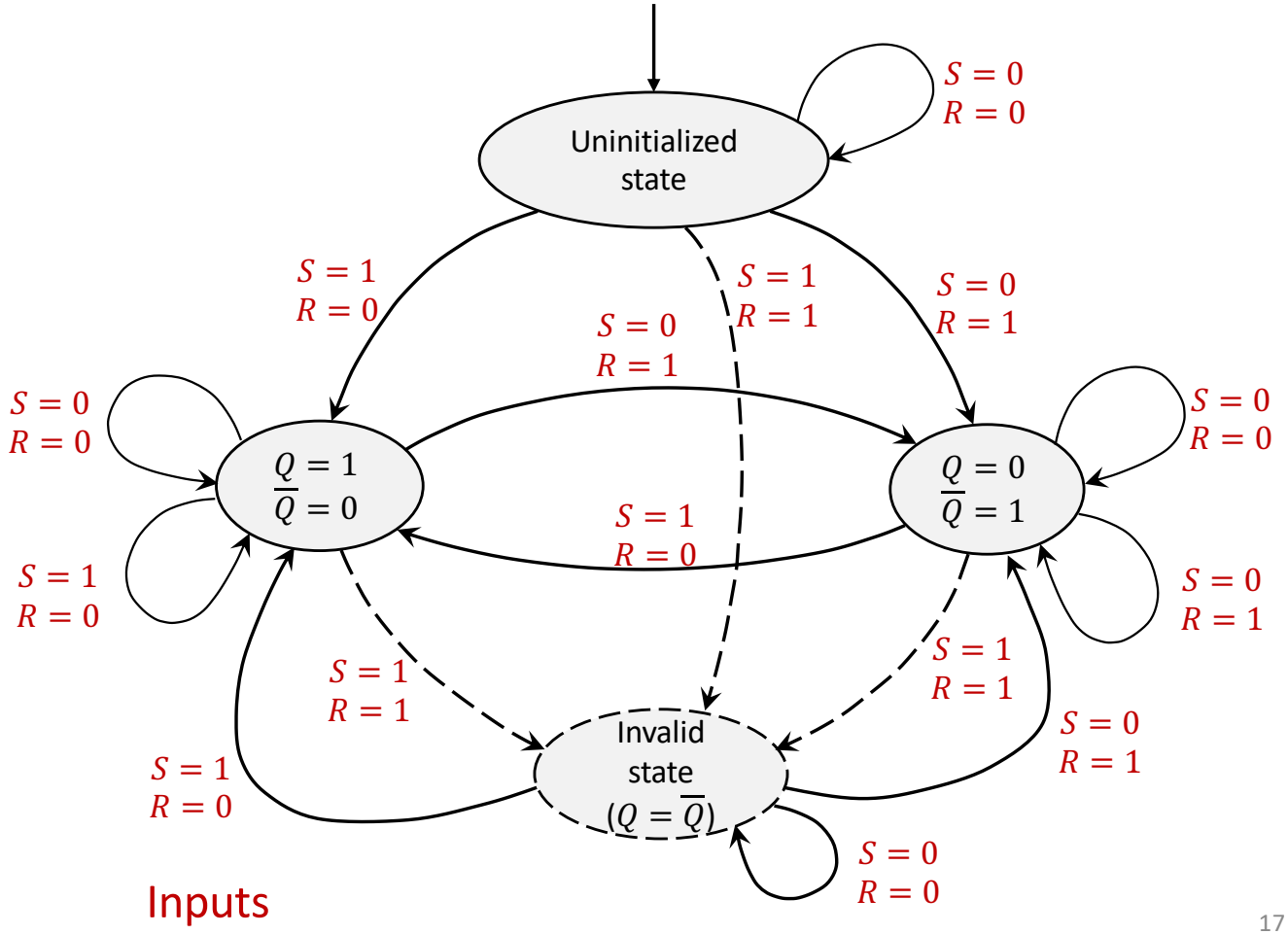
Finite State Machine (FSM) (or Finite State Automaton, or simply State Machine)

FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;

b) Has one initial state;

c) It gets inputs from outside;



Finite State Machine (FSM) (or Finite State Automaton, or simply State Machine)

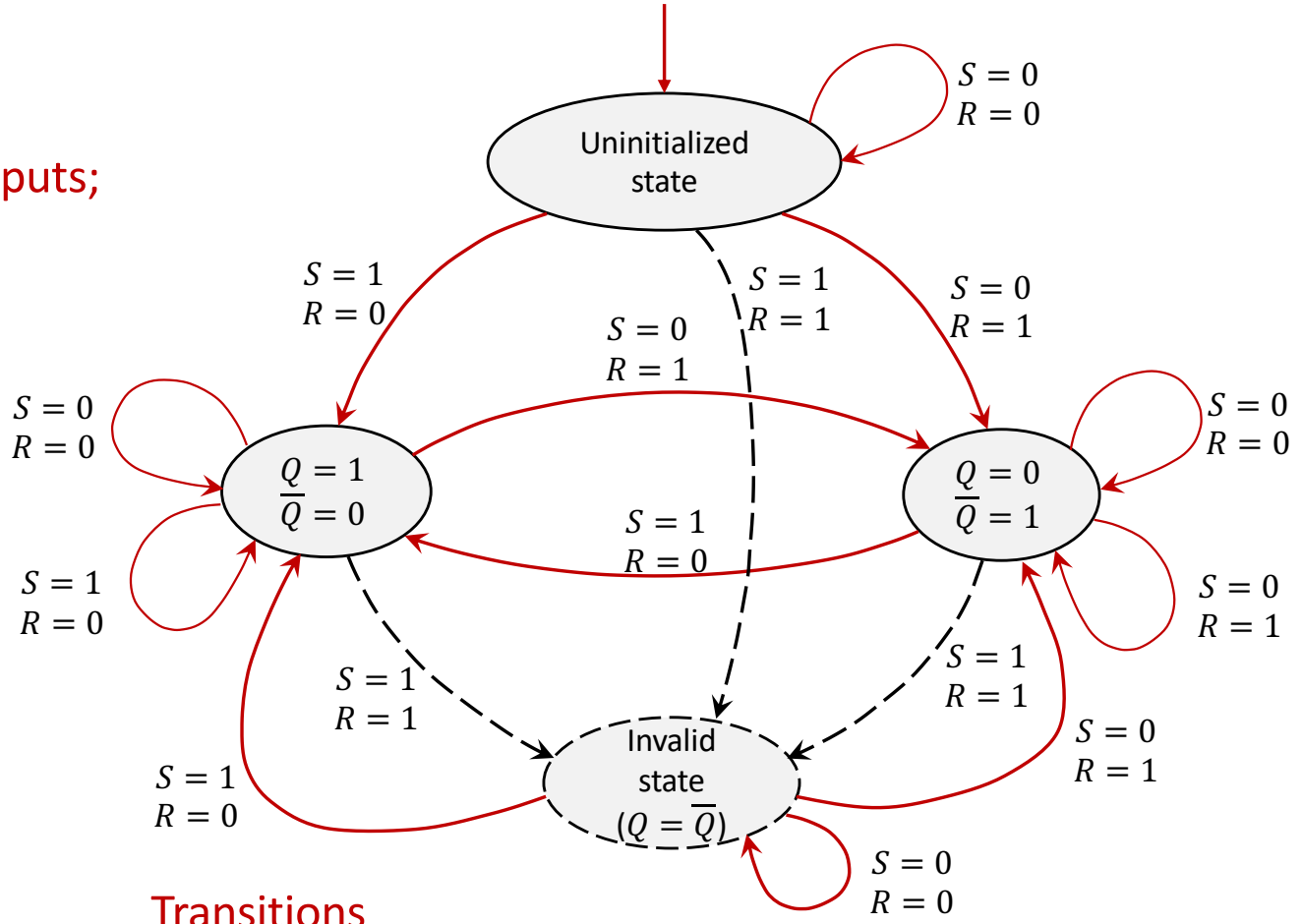
FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;

b) Has one initial state;

c) It gets inputs from outside;

d) It changes its state in response to inputs;

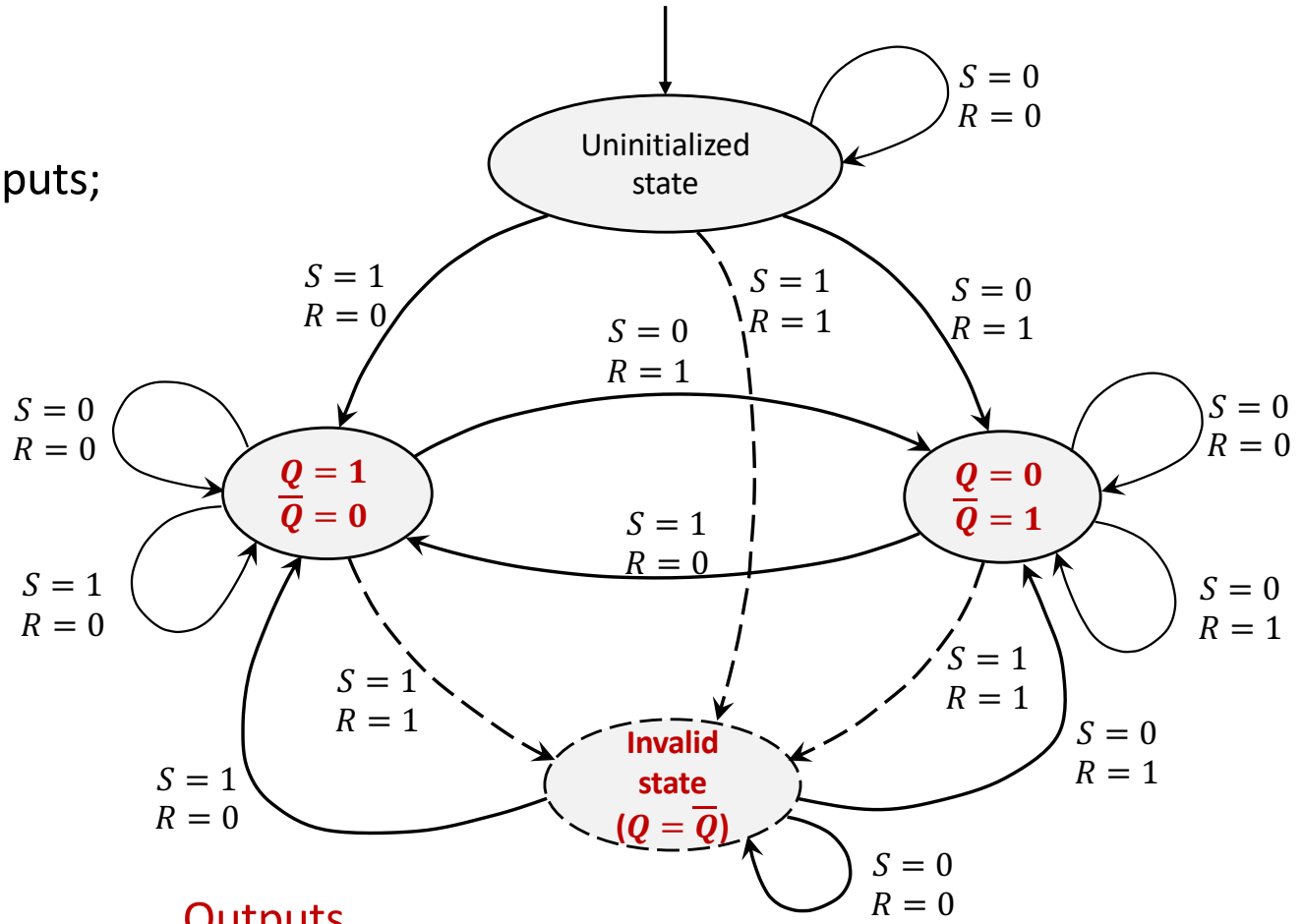


Transitions

Finite State Machine (FSM) (or Finite State Automaton, or simply State Machine)

FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;
b) Has one initial state;
c) It gets inputs from outside
d) It changes its state in response to inputs;
e) It can produce output as well



Outputs

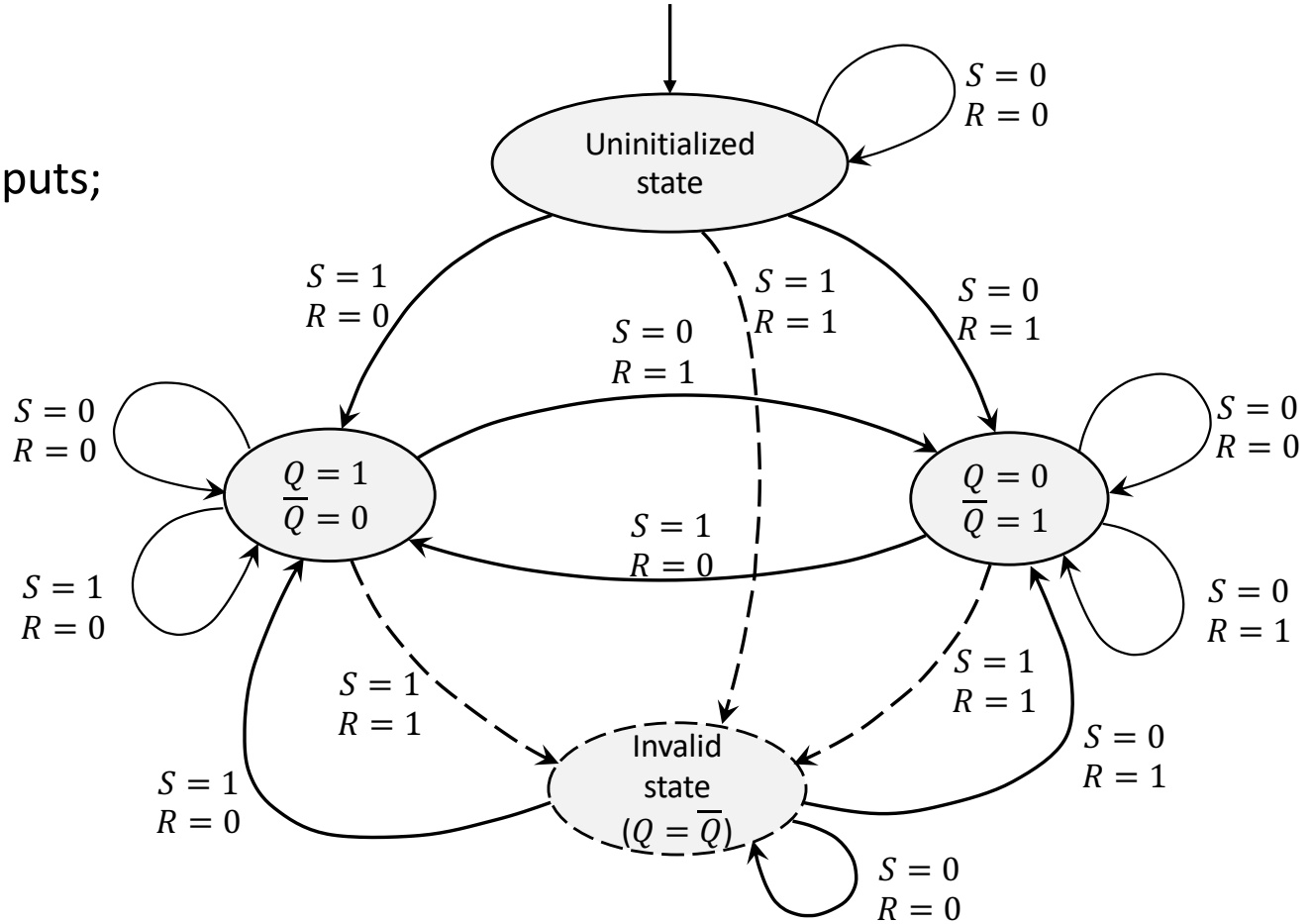
Finite State Machine (FSM)

FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;
b) Has one initial state;
c) It gets inputs from outside
d) It changes its state in response to inputs;
e) It can produce output as well

This FSM is deterministic

(There is no state with two outgoing transitions for the same input)



Finite State Machine (FSM)

FSM - an abstract machine, defined as follows:

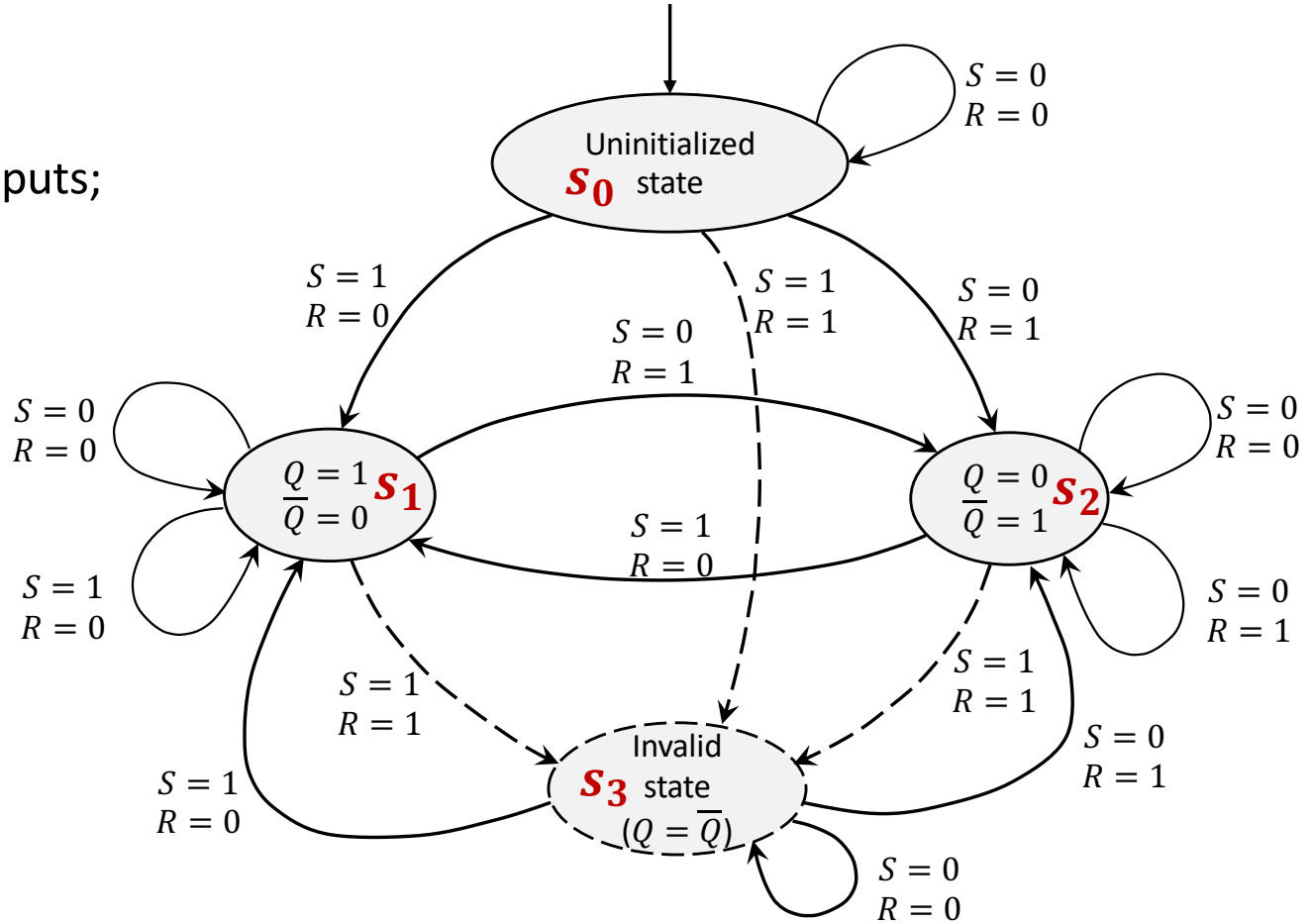
a) It can be in exactly one of a finite number of states at any given time;
b) Has one initial state;
c) It gets inputs from outside
d) It changes its state in response to inputs;
e) It can produce output as well

This FSM is deterministic

(There is no state with two outgoing transitions for the same input)

Formalization:

$S = \{s_0, s_1, \dots, s_n\}$ - set of states



Finite State Machine (FSM)

FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;
b) Has one initial state;
c) It gets inputs from outside
d) It changes its state in response to inputs;
e) It can produce output as well

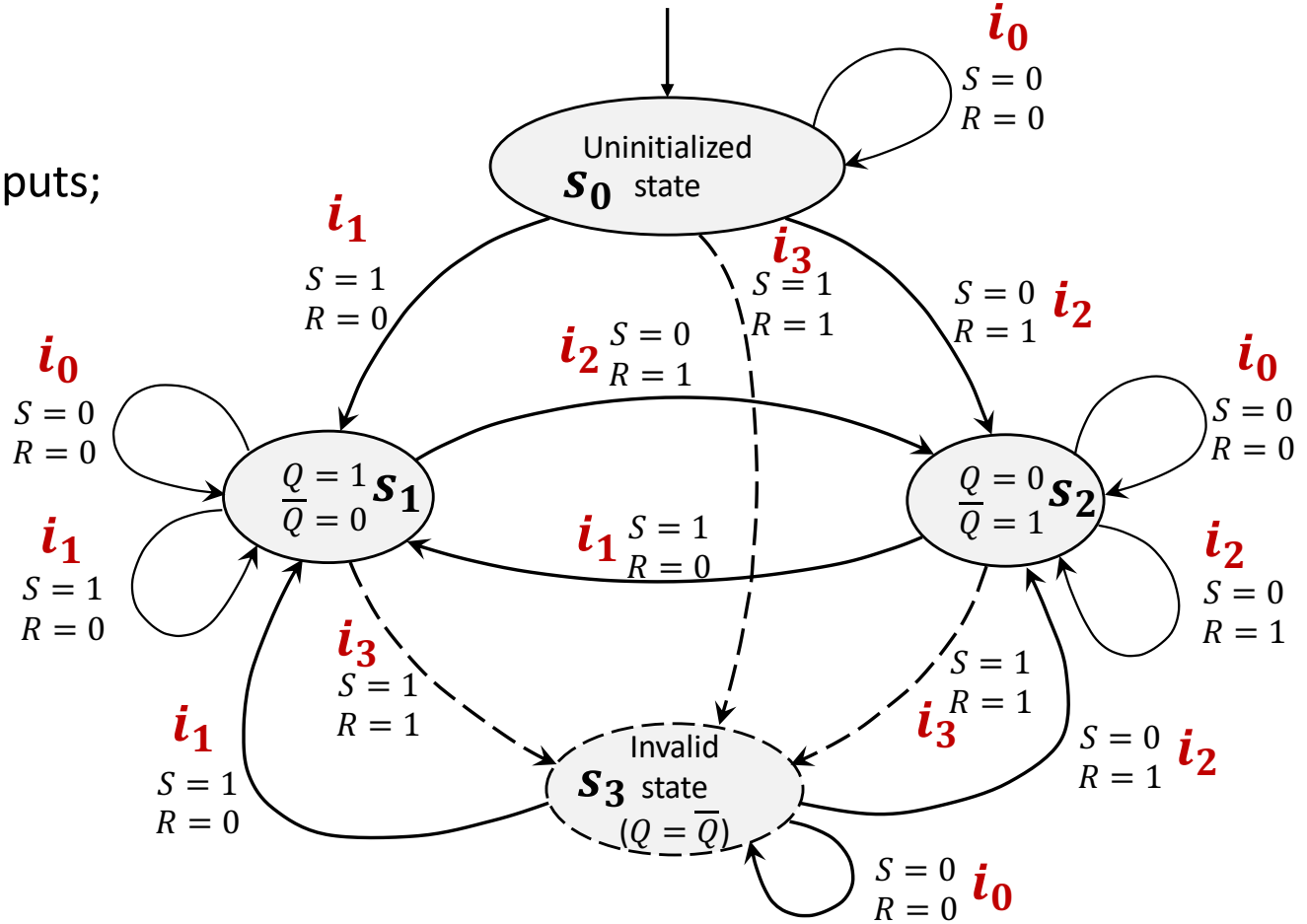
This FSM is deterministic

(There is no state with two outgoing transitions for the same input)

Formalization:

$S = \{s_0, s_1, \dots, s_n\}$ - set of states

$I = \{i_0, i_1, \dots, i_k\}$ - inputs



Finite State Machine (FSM)

FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;
b) Has one initial state;
c) It gets inputs from outside
d) It changes its state in response to inputs;
e) It can produce output as well

This FSM is deterministic

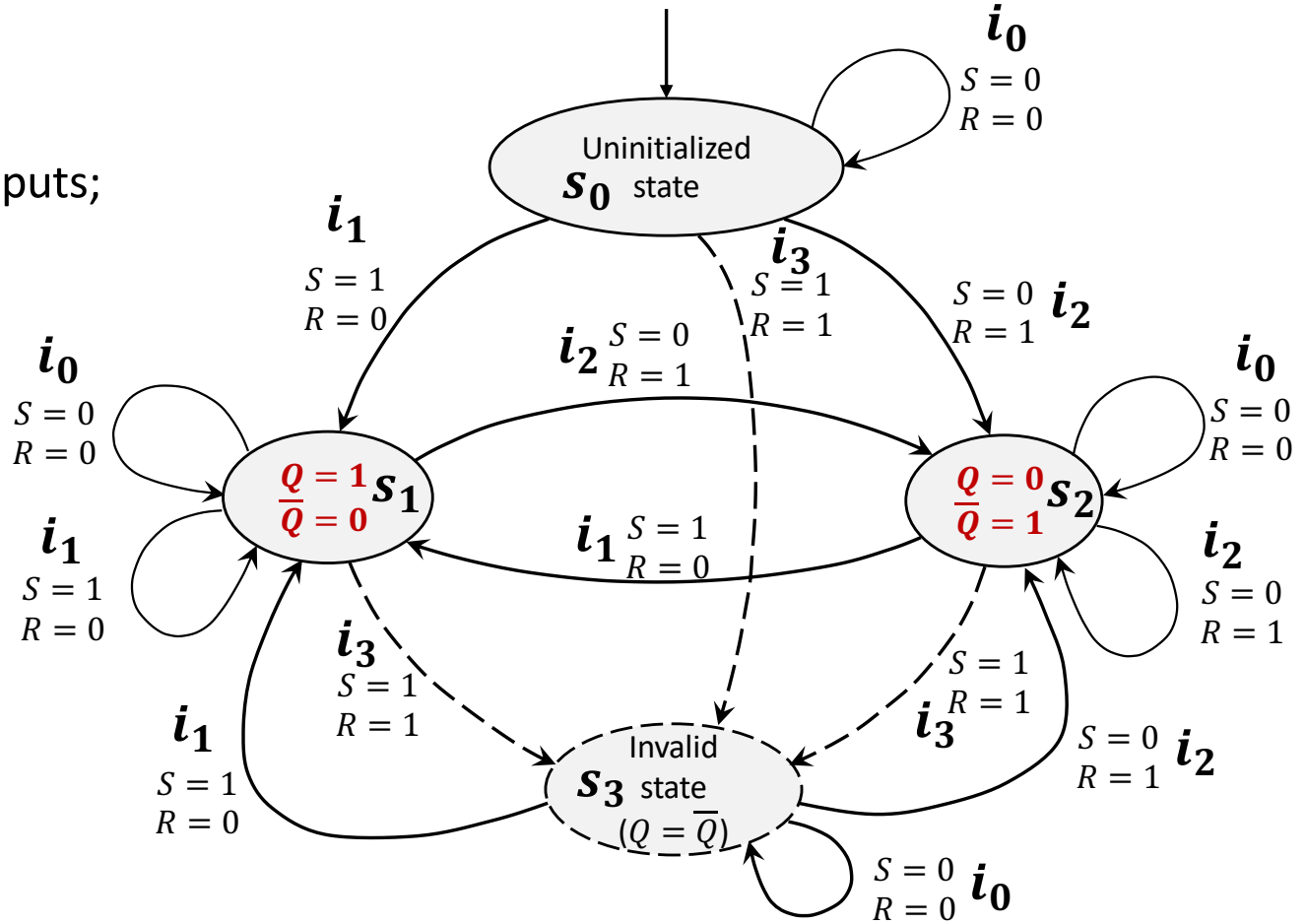
(There is no state with two outgoing transitions for the same input)

Formalization:

$S = \{s_0, s_1, \dots, s_n\}$ - set of states

$I = \{i_0, i_1, \dots, i_k\}$ - inputs

$O = \{o_0, o_1, \dots, o_m\}$ - outputs



Finite State Machine (FSM)

FSM - an abstract machine, defined as follows:

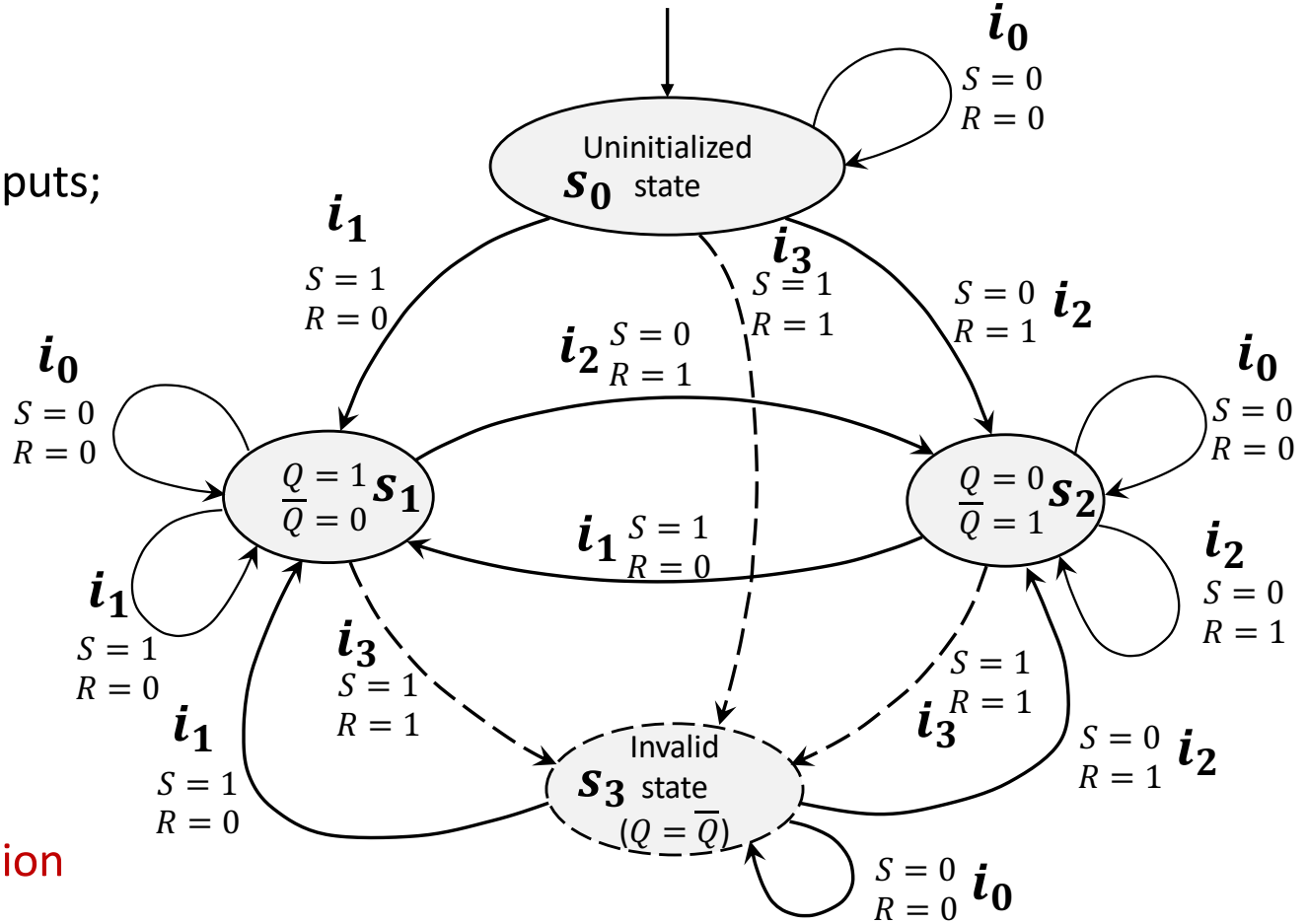
a) It can be in exactly one of a finite number of states at any given time;
b) Has one initial state;
c) It gets inputs from outside
d) It changes its state in response to inputs;
e) It can produce output as well

This FSM is deterministic

(There is no state with two outgoing transitions for the same input)

Formalization:

$S = \{s_0, s_1, \dots, s_n\}$ - set of states
$I = \{i_0, i_1, \dots, i_k\}$ - inputs
$O = \{o_0, o_1, \dots, o_m\}$ - outputs
$F = (S \times I) \to (S \times O)$ – transition function



Finite State Machine (FSM)

FSM - an abstract machine, defined as follows:

a) It can be in exactly one of a finite number of states at any given time;
b) Has one initial state;
c) It gets inputs from outside
d) It changes its state in response to inputs;
e) It can produce output as well

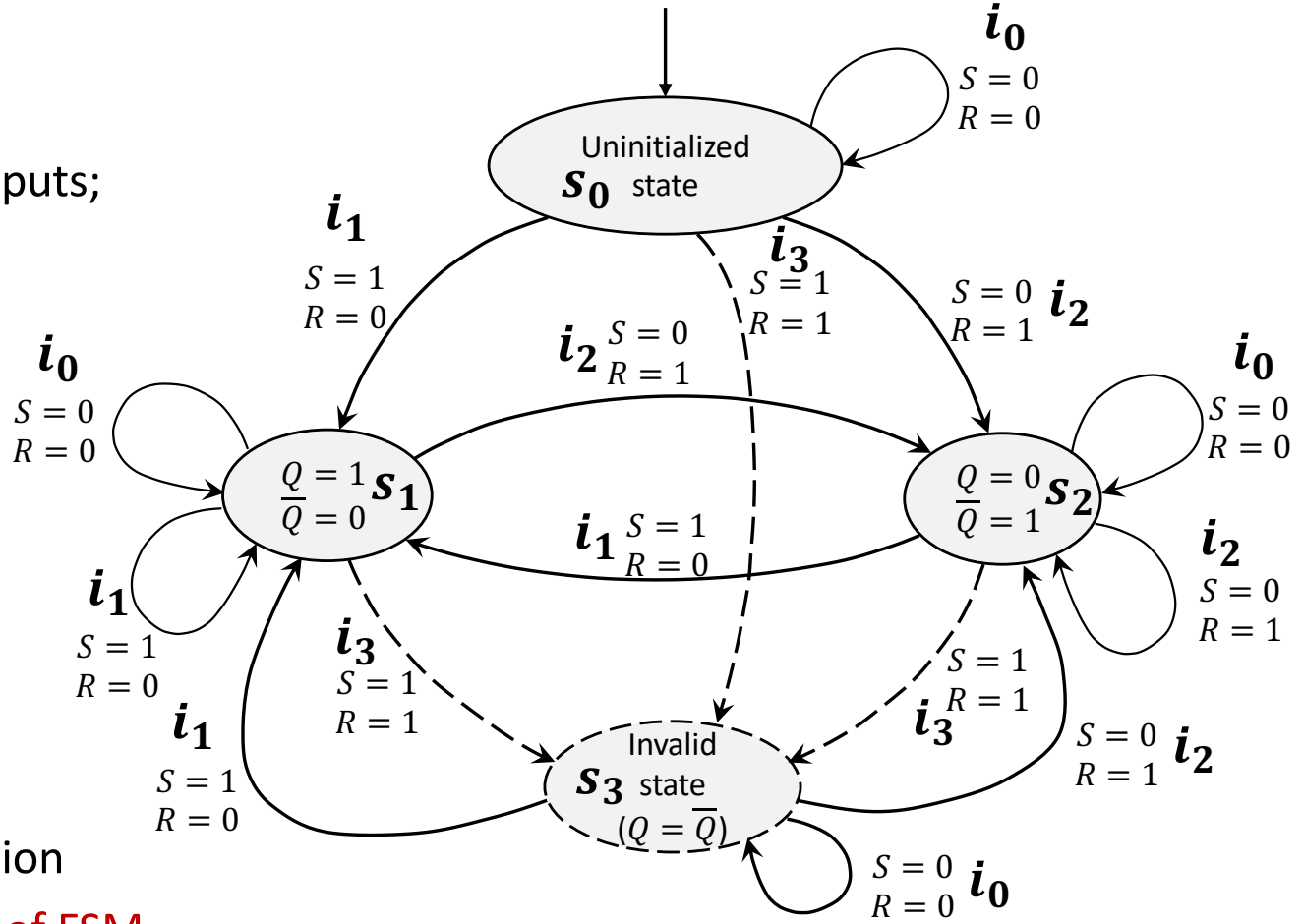
This FSM is deterministic

(There is no state with two outgoing transitions for the same input)

Formalization:

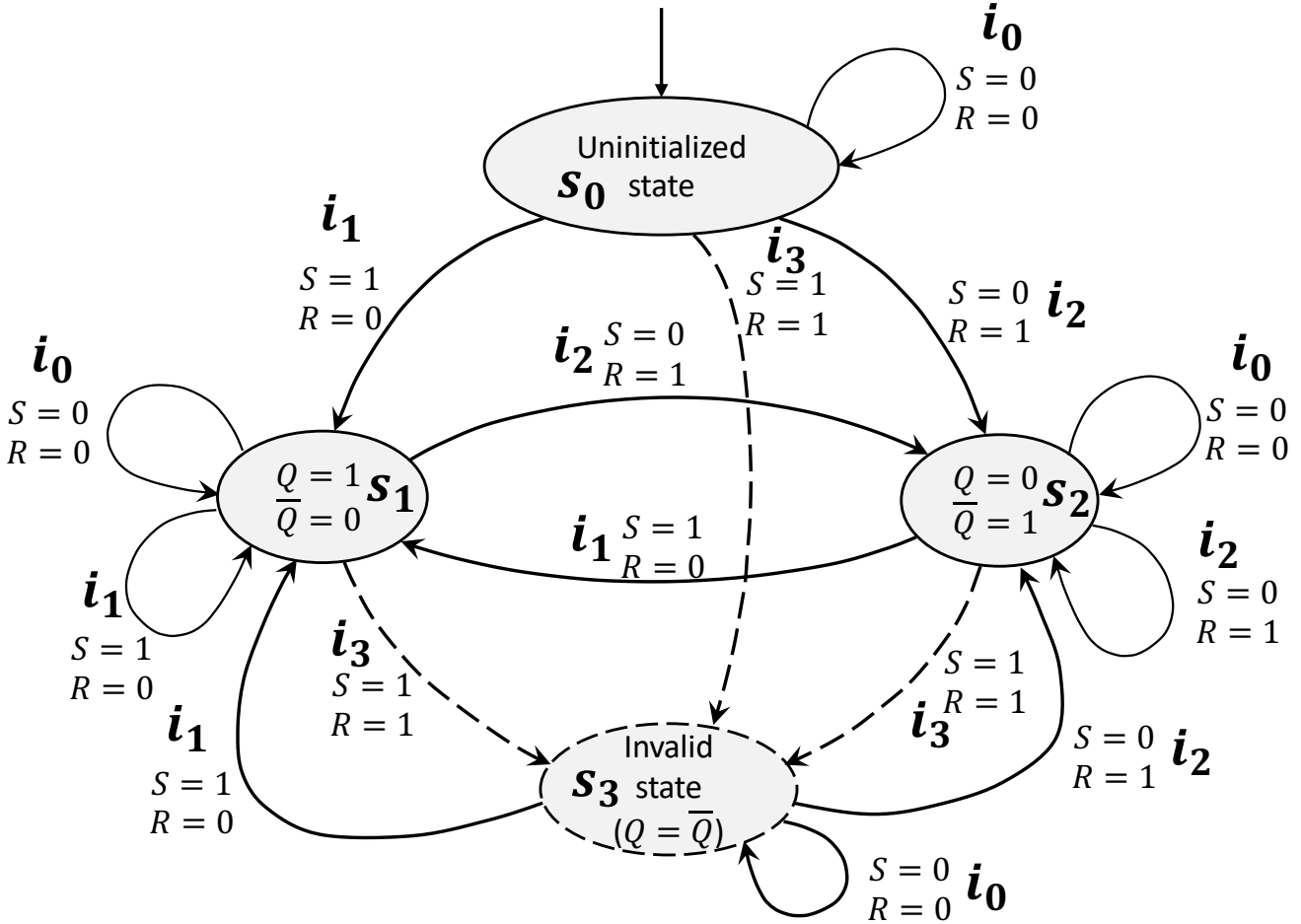
$S = \{s_0, s_1, \dots, s_n\}$ - set of states
$I = \{i_0, i_1, \dots, i_k\}$ - inputs
$O = \{o_0, o_1, \dots, o_m\}$ - outputs
$F = (S \times I) \rightarrow (S \times O)$ – transition function

State transition diagram – visualization of FSM



Finite State Machine (FSM) Visualization

State transition diagram

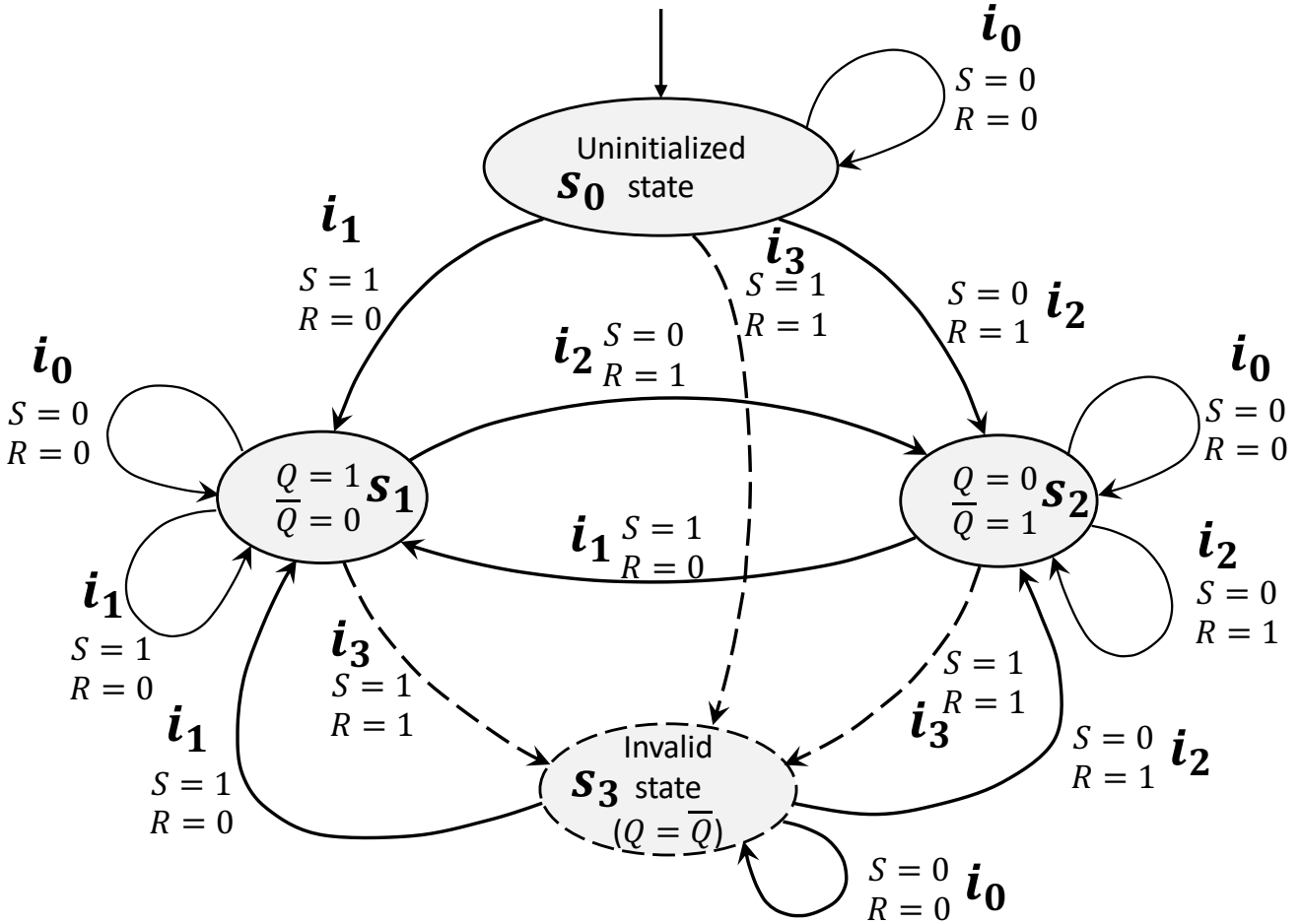


Finite State Machine (FSM) Visualization

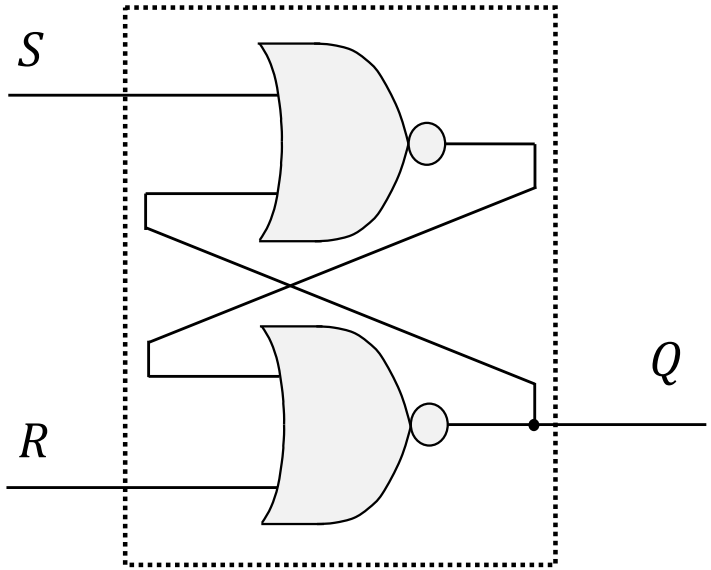
Transition table – an alternative

Current state	Input	New state	Output
s_0	i_0	s_0	undefined
s_0	i_1	s_1	1
s_0	i_2	s_2	0
…			

State transition diagram



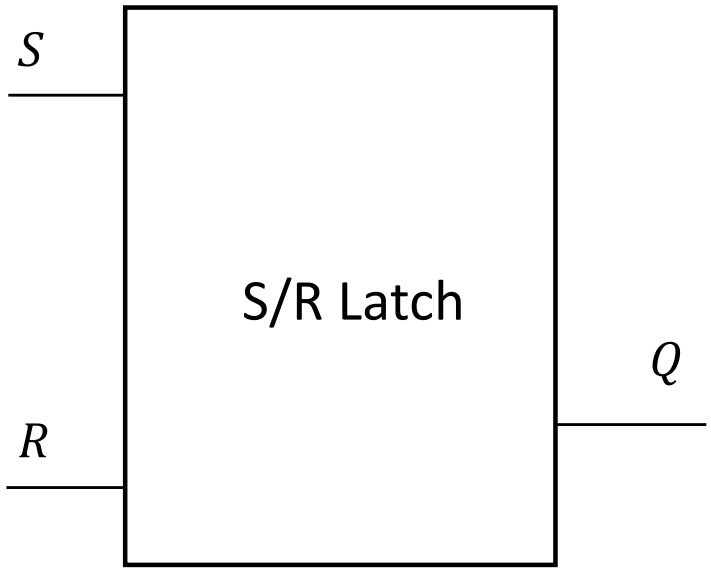
Recap: Transparent S/R Latch



Set/Reset Latch
(or transparent latch)

S	***R***	***Q***
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal inputs

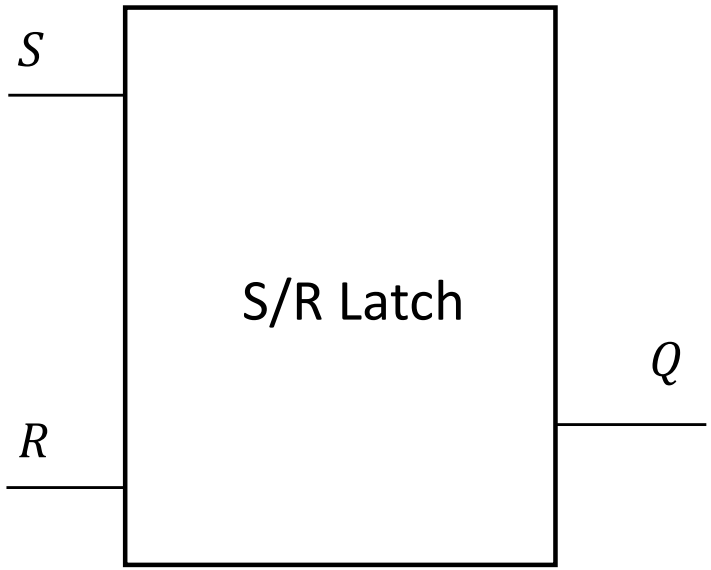
S/R Latch: A Less Detailed Representation



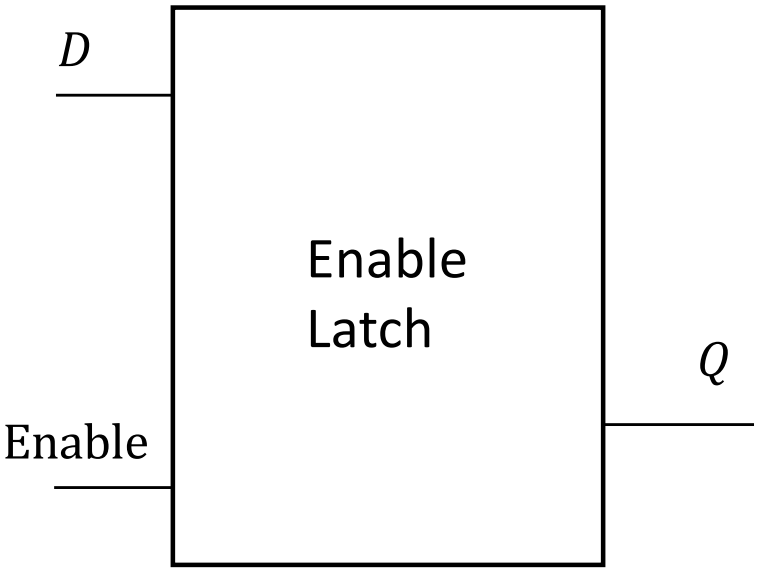
Set/Reset Latch
(or transparent latch)

S	***R***	***Q***
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal inputs

Enable Latch: An Alternative to S/R Latch

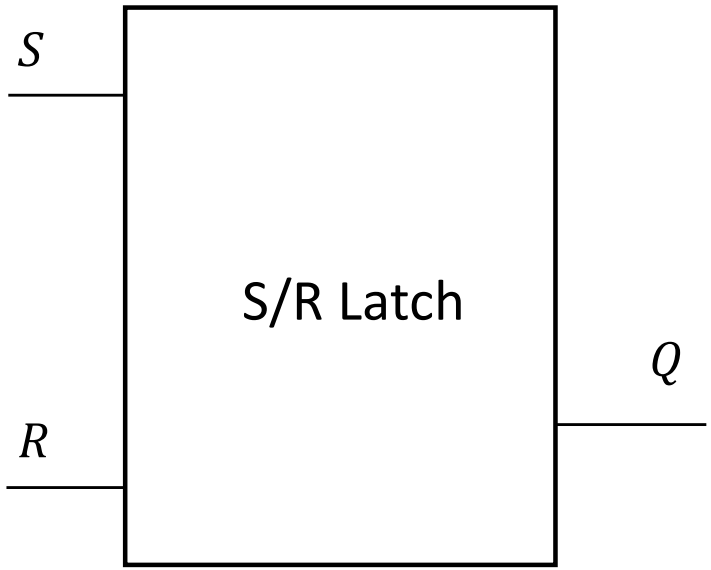


Set/Reset Latch
(or transparent latch)

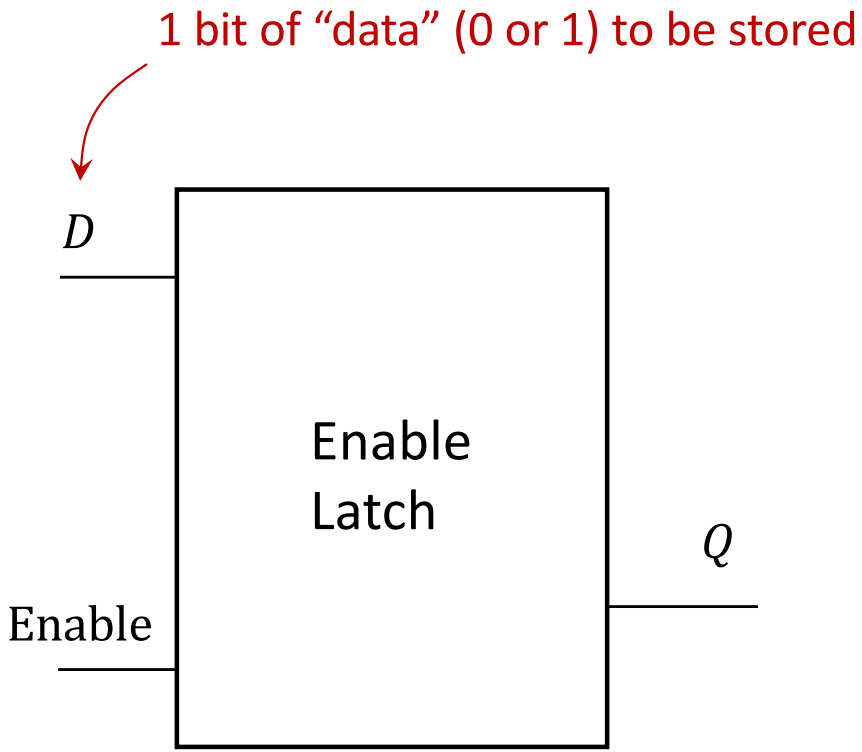


Enable Latch

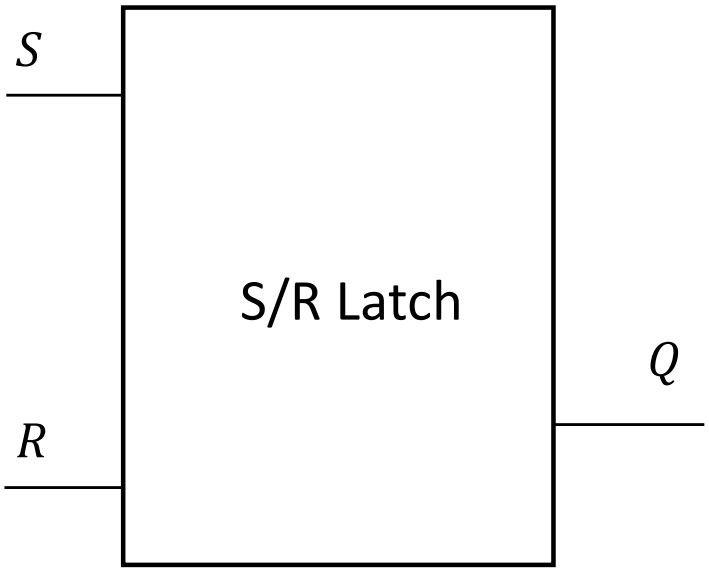
Enable Latch: An Alternative to S/R Latch



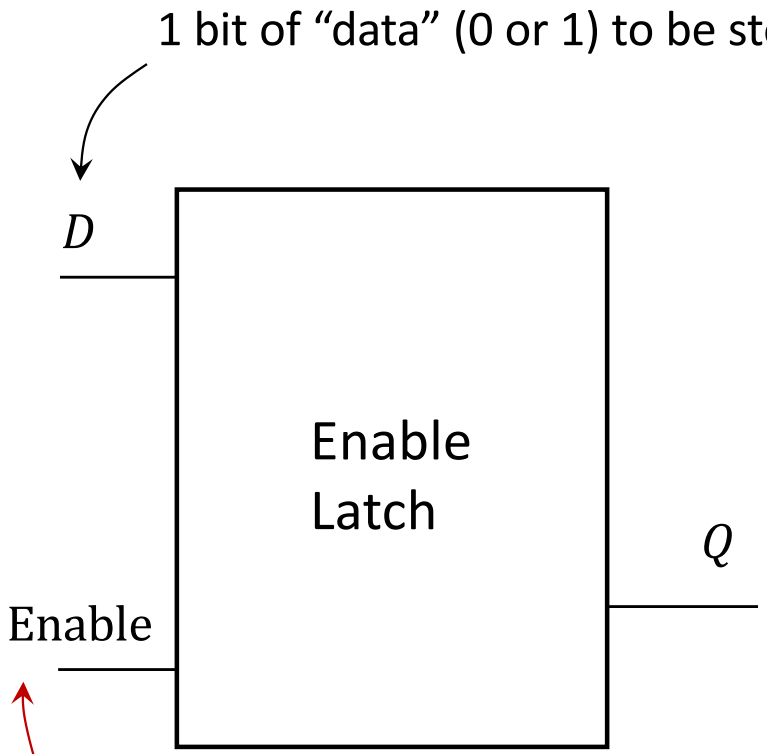
Set/Reset Latch
(or transparent latch)



Enable Latch: An Alternative to S/R Latch



Set/Reset Latch
(or transparent latch)



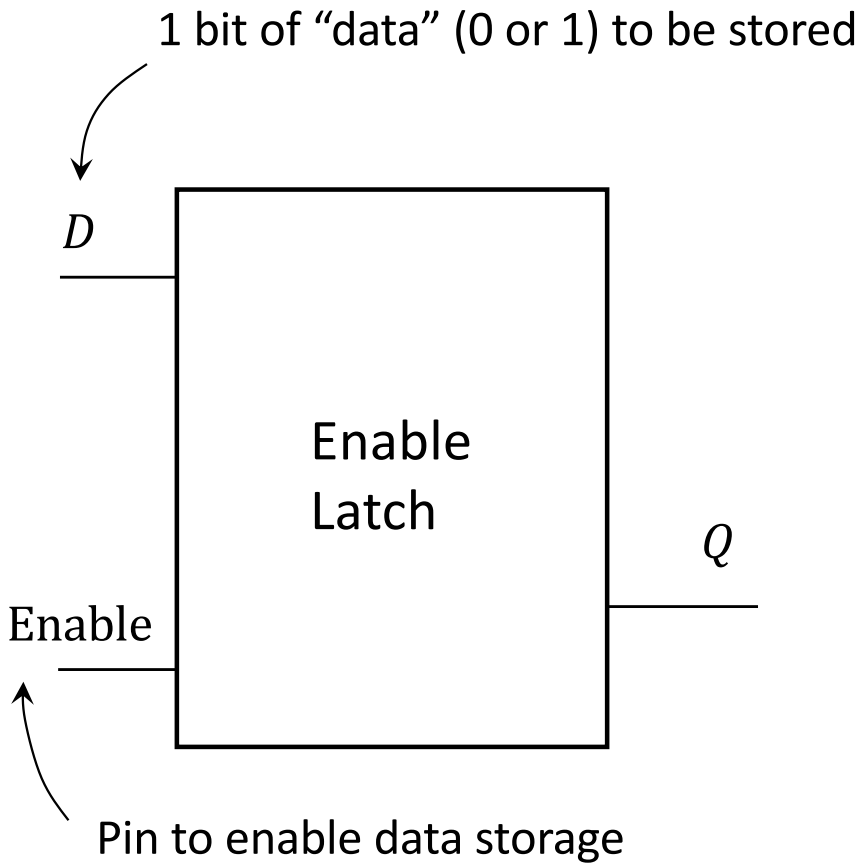
Pin to enable data storage:

- if set to “1”, data is stored;
- If set to “0”, data is ignored

Enable Latch: An Alternative to S/R Latch

Advantage:

- No need to care about an invalid input combination (like S=1, R=1 is prohibited for S/R latch);
- All input combinations allowed
- Simpler usage

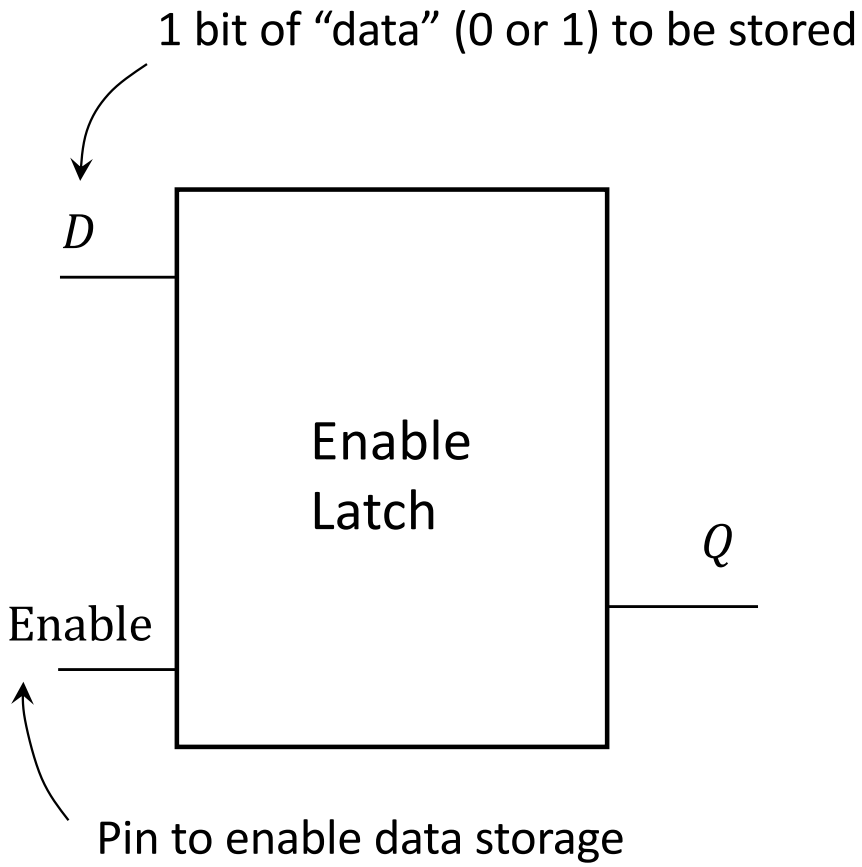


Enable Latch: An Alternative to S/R Latch

Advantage:

- No need to care about invalid input combination (like S=1, R=1 is prohibited for S/R latch);
- All input combinations allowed
- Simpler usage

Data is stored by latch, until a different value “D” arrives, at a time when pin “Enable” is activated

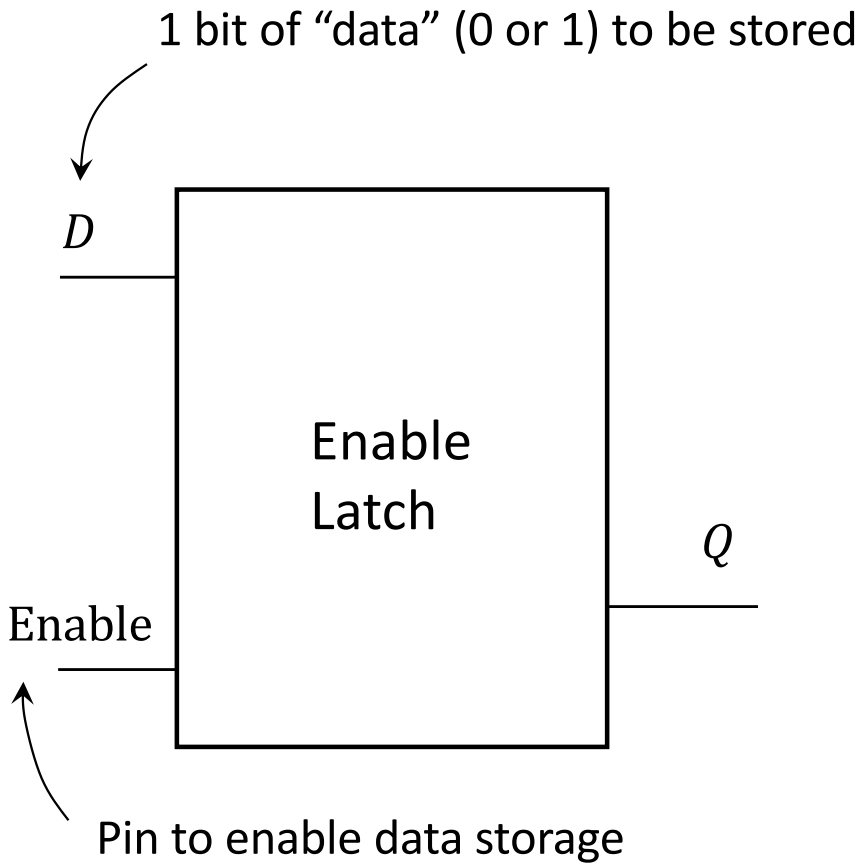


Enable Latch: An Alternative to S/R Latch

Advantage:

- No need to care about invalid input combination (like S=1, R=1 is prohibited for S/R latch);
- All input combinations allowed
- Simpler usage

Data is stored by latch, until a different value “D” arrives, at a time when pin “Enable” is activated



Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger		
Clock Presence		
Reliability (for result correctness)		
Memory Element Presence		
Operation Speed		
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	
Clock Presence		
Reliability (for result correctness)		
Memory Element Presence		
Operation Speed		
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence		
Reliability (for result correctness)		
Memory Element Presence		
Operation Speed		
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence		Clock governs the entire circuit activity
Reliability (for result correctness)		
Memory Element Presence		
Operation Speed		
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)		
Memory Element Presence		
Operation Speed		
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)		Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence		
Operation Speed		
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence		
Operation Speed		
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence		Used
Operation Speed		
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence	Not used	Used
Operation Speed		
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence	Not used	Used
Operation Speed	Faster (no clock signal for synchronisation delay is used	
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence	Not used	Used
Operation Speed	Faster (no clock signal for synchronisation delay is used	Slower (due to overpessimistic synchronisation delays)
Power Consumption		
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence	Not used	Used
Operation Speed	Faster (no clock signal for synchronisation delay is used	Slower (due to overpessimistic synchronisation delays)
Power Consumption	Lower	
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence	Not used	Used
Operation Speed	Faster (no clock signal for synchronisation delay is used	Slower (due to overpessimistic synchronisation delays)
Power Consumption	Lower	Higher (e.g. due to the presence of flip-flops, consuming power for data storage)
Logical Complexity, Size (the number of elements)		
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence	Not used	Used
Operation Speed	Faster (no clock signal for synchronisation delay is used	Slower (due to overpessimistic synchronisation delays)
Power Consumption	Lower	Higher (e.g. due to the presence of flip-flops, consuming power for data storage)
Logical Complexity, Size (the number of elements)	Simpler, smaller (*Note:* an asynchronous circuit might behaive as a synchronous, but at the price of a higher hardware implementation complexity)	
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence	Not used	Used
Operation Speed	Faster (no clock signal for synchronisation delay is used	Slower (due to overpessimistic synchronisation delays)
Power Consumption	Lower	Higher (e.g. due to the presence of flip-flops, consuming power for data storage)
Logical Complexity, Size (the number of elements)	Simpler, smaller (*Note:* an asynchronous circuit might behave as a synchronous, but at the price of a higher hardware implementation complexity)	More complex, larger
Sample Use Cases		

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence	Not used	Used
Operation Speed	Faster (no clock signal for synchronisation delay is used	Slower (due to overpessimistic synchronisation delays)
Power Consumption	Lower	Higher (e.g. due to the presence of flip-flops, consuming power for data storage)
Logical Complexity, Size (the number of elements)	Simpler, smaller (*Note:* an asynchronous circuit might behave as a synchronous, but at the price of a higher hardware implementation complexity)	More complex, larger
Sample Use Cases		Register files; Most of the circuits containing memory elements

Synchronous vs. Asynchronous Circuits

Characteristic	**Asynchronous** (like S/R latch)	**Synchronous** (like Flip-Flop)
Output Update Trigger	Any change of one of the inputs	A clock signal, together with other input signals
Clock Presence	No clock present	Clock governs the entire circuit activity
Reliability (for result correctness)	Less reliable (prone to incorrect result, subject to propagation delays)	Reliable (guarantees a correct result, independently of propagation delays)
Memory Element Presence	Not used	Used
Operation Speed	Faster (no clock signal for synchronisation delay is used)	Slower (due to overpessimistic synchronisation delays)
Power Consumption	Lower	Higher (e.g. due to the presence of flip-flops, consuming power for data storage)
Logical Complexity, Size (the number of elements)	Simpler, smaller (*Note:* an asynchronous circuit might behave as a synchronous, but at the price of a higher hardware implementation complexity)	More complex, larger
Sample Use Cases	Arithmetic-Logic Unit (ALU); Small fast peripheral circuits supporing CPU operation	Register files; Most of the circuits containing memory elements

Digital Circuits Classification