Computer Architecture
Computer Engineering Track
Tutorial 08

The Revision of Selected Topics II:

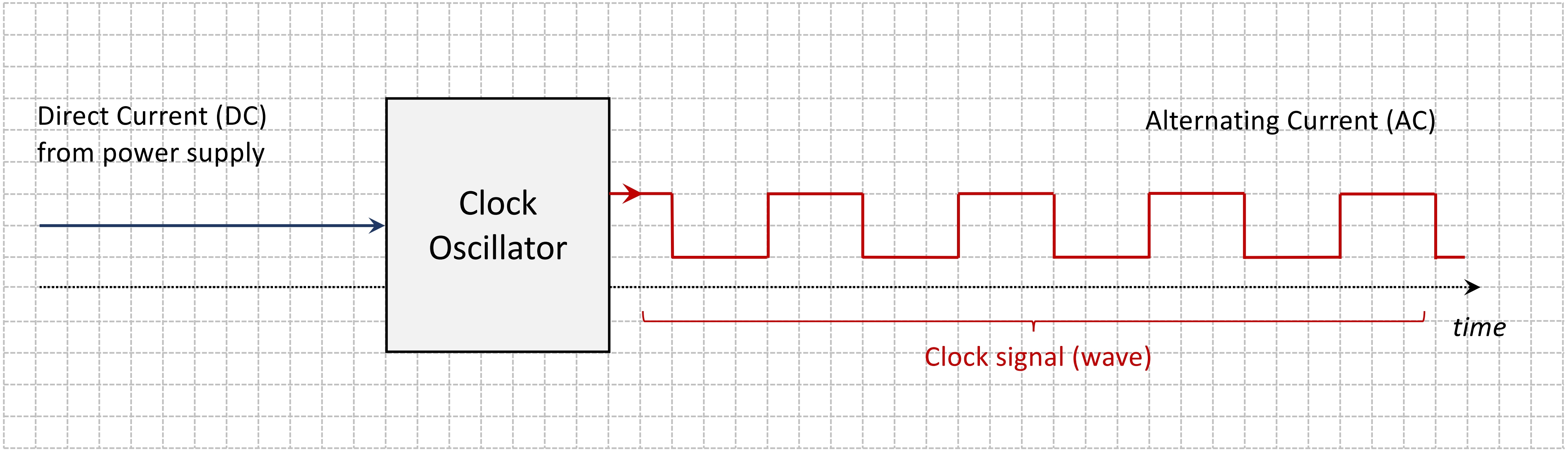
Latches and Flip-Flops
Synchronous and Asynchronous Circuits

Artem Burmyakov, Muhammad Fahim, Alexander Tormasov

October 22, 2020

Clock Generator (Oscillator)

An electronic device, producing a periodic clock signal

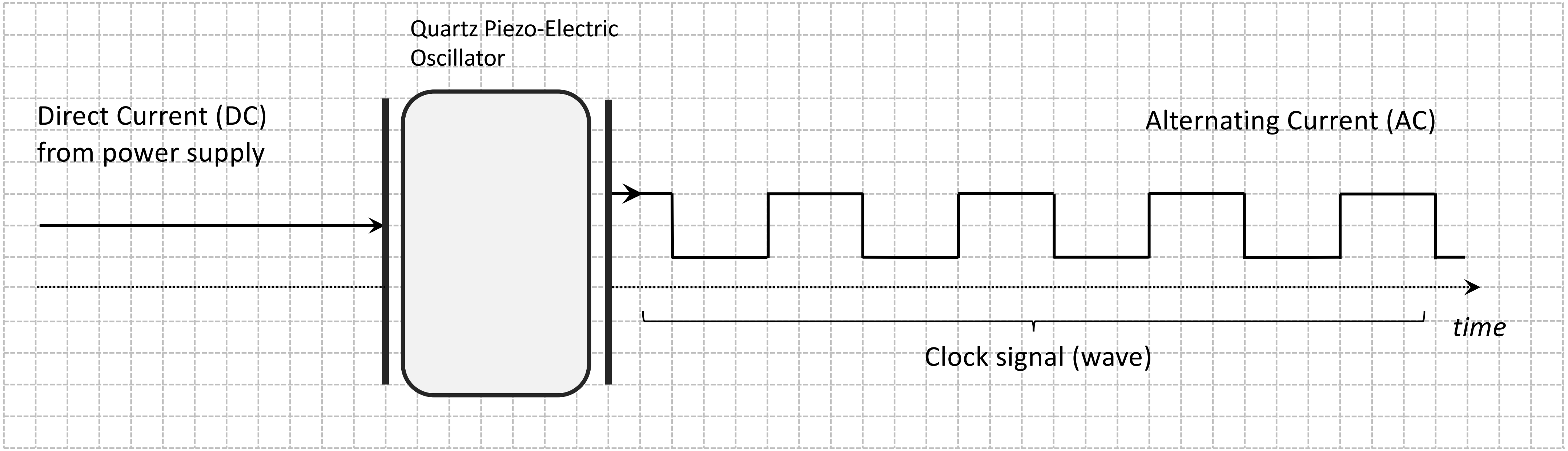


Clock oscillator – a physical device, that converts direct current (DC) from a power supply into an alternating periodic signal

Oscillation – a repetitive variation, typically in time, of some measure between two or several values

Clock Generator (Oscillator)

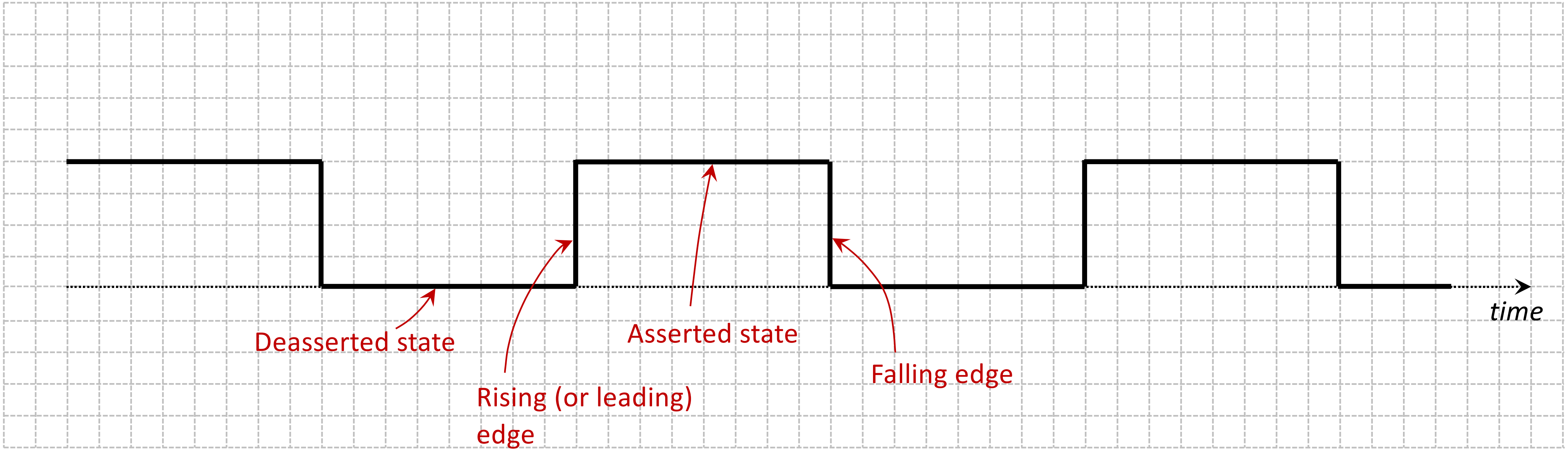
An electronic device, producing a periodic clock signal


Quartz Piezo-Electric Oscillator
Direct Current (DC) from power supply
Alternating Current (AC)
time
Clock signal (wave)

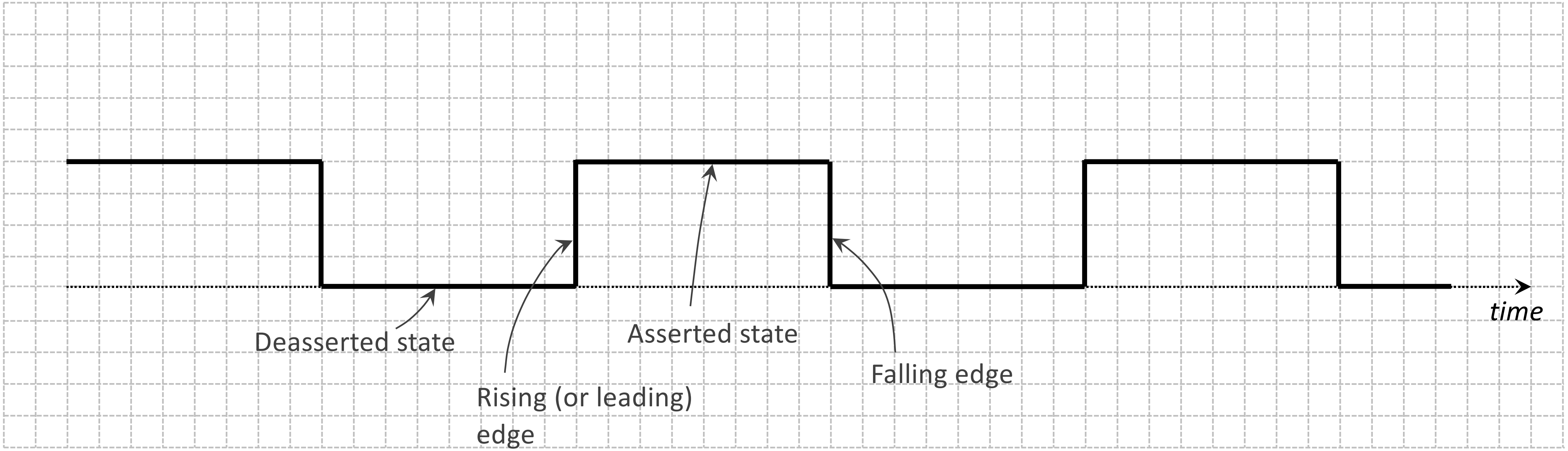
Clock oscillator – a physical device, that converts direct current (DC) from a power supply into an alternating periodic signal

Oscillation – a repetitive variation, typically in time, of some measure between two or several values

Clock Signal Terminology



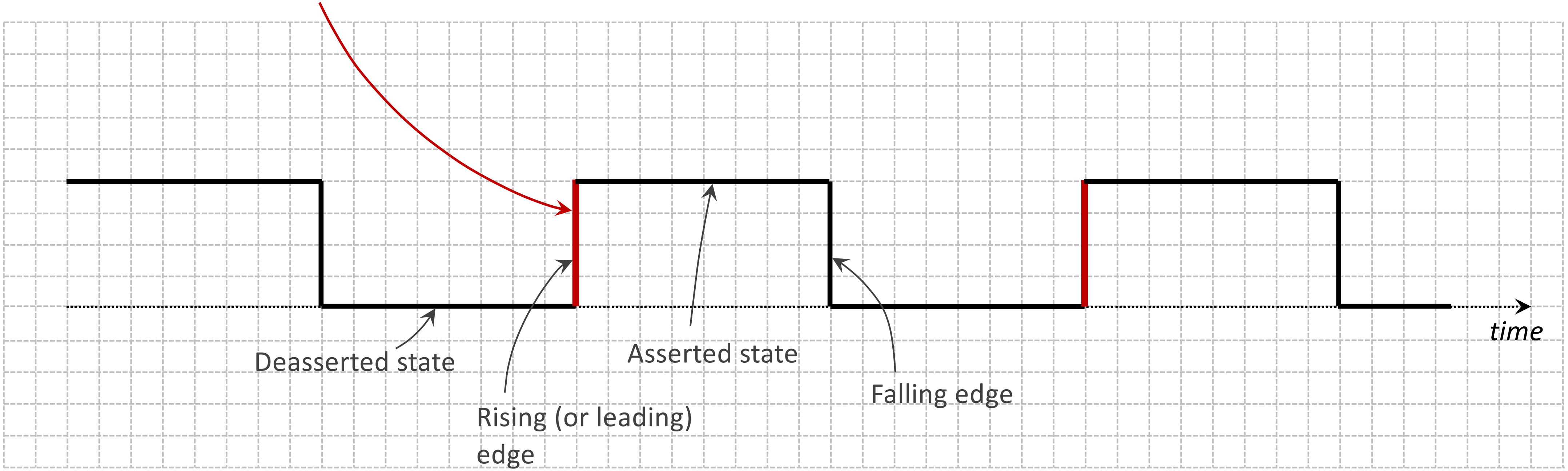
Edge-Triggered and Level-Triggered Timing Methodologies



Edge-Triggered and Level-Triggered Timing Methodologies

Edge-Triggered:

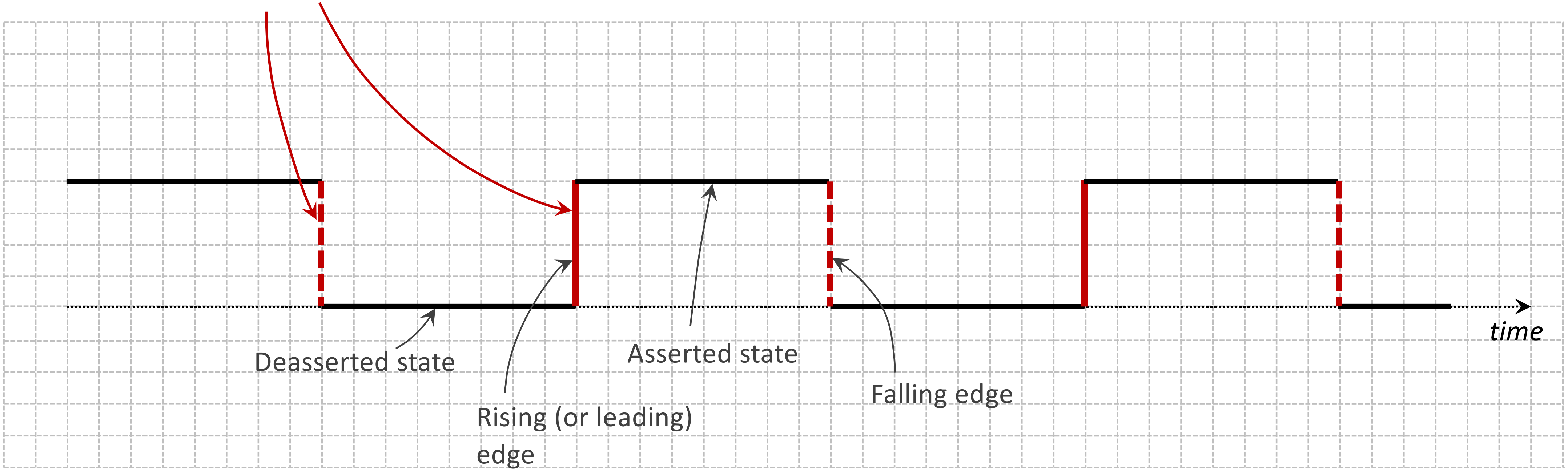
a circuit is activated either on a raising or a falling edge



Edge-Triggered and Level-Triggered Timing Methodologies

Edge-Triggered:

a circuit is activated either on a raising or a falling edge



Deasserted state

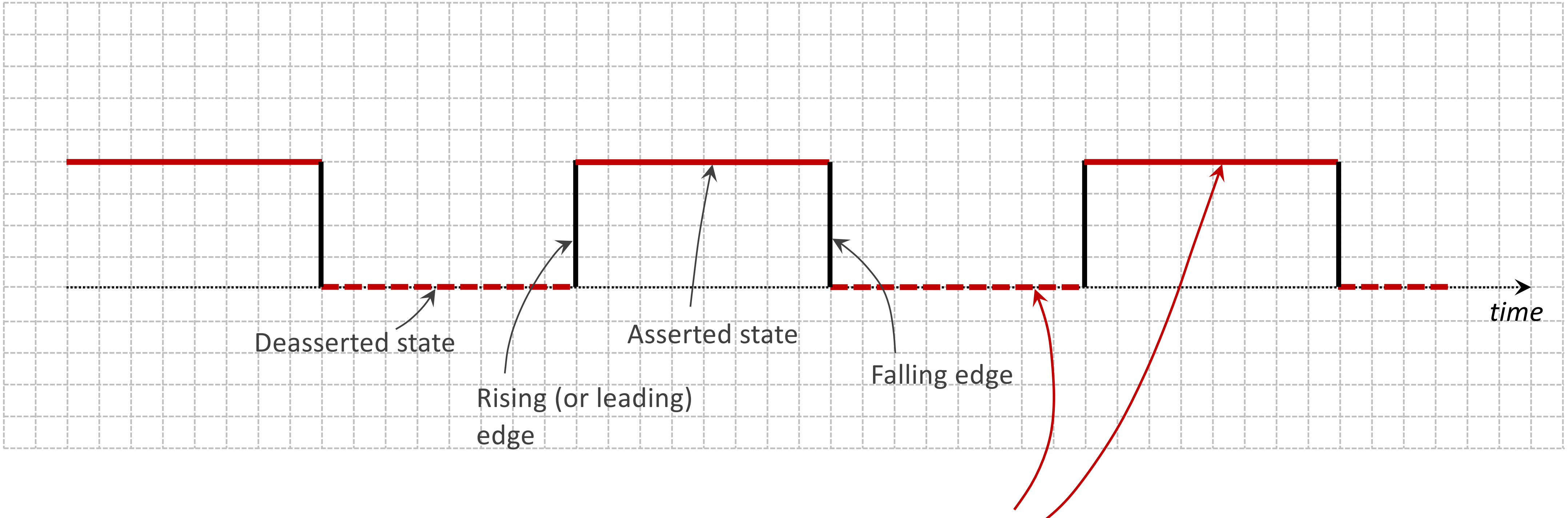
Asserted state

Rising (or leading) edge

Falling edge

time

Edge-Triggered and Level-Triggered Timing Methodologies



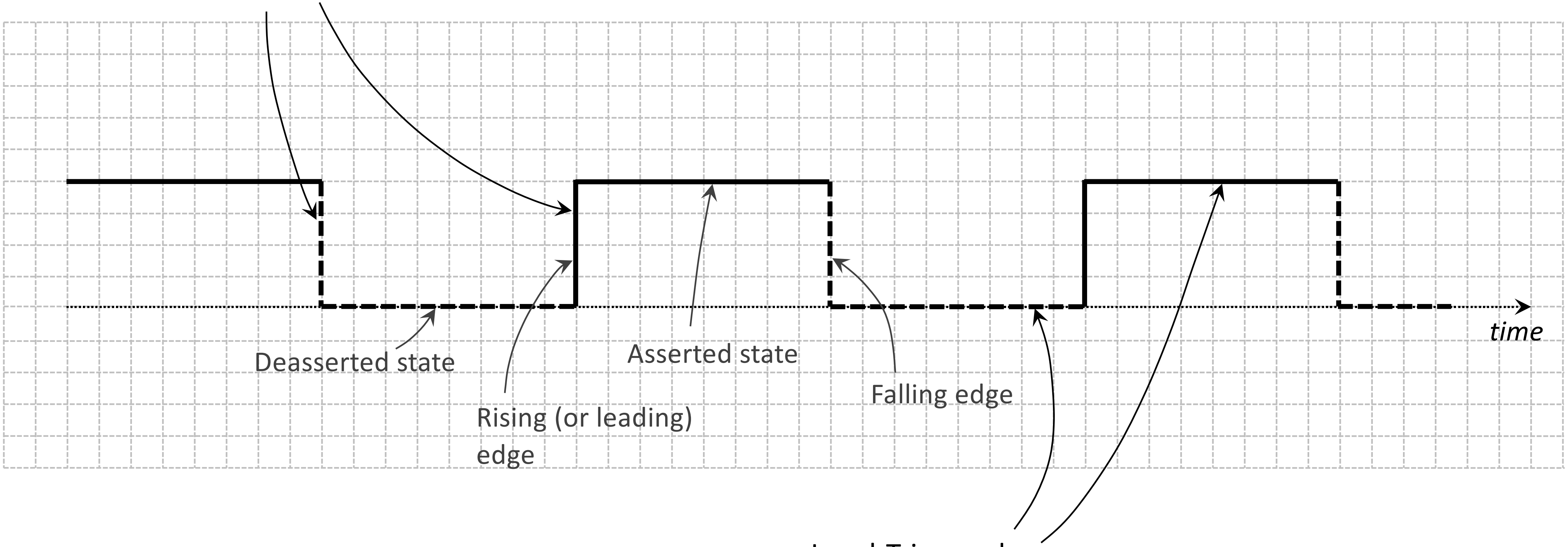
Level-Triggered:

a circuit is activated either when clock signal is asserted, or deasserted

Edge-Triggered and Level-Triggered Timing Methodologies

Edge-Triggered:

a circuit is activated either on a raising or a falling edge



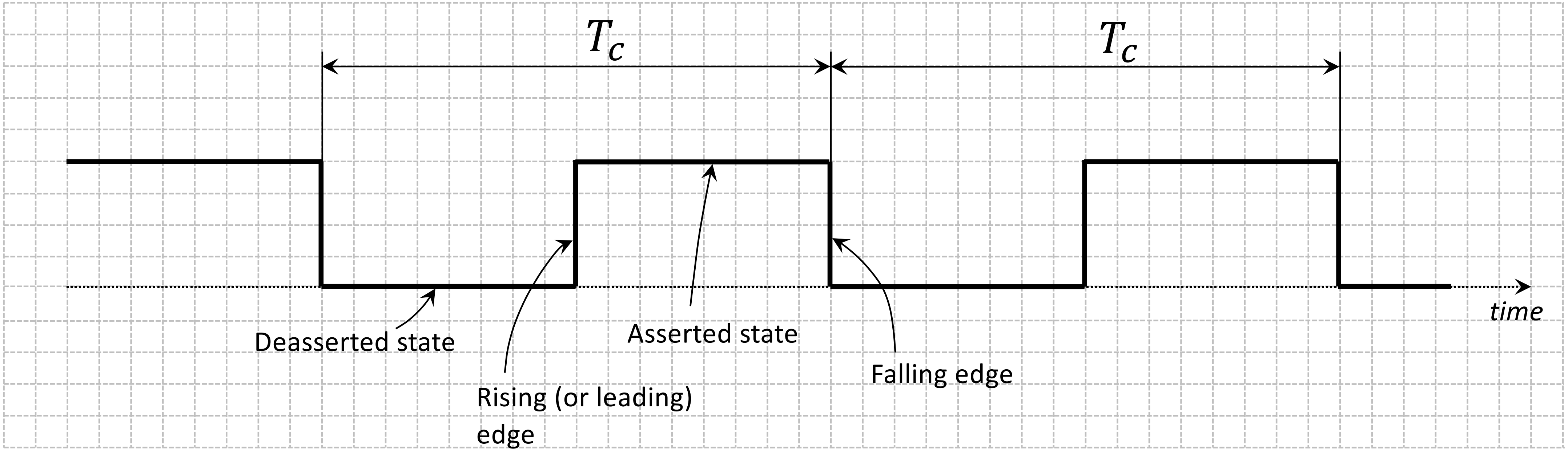
Level-Triggered:

a circuit is activated either when clock signal is asserted, or deasserted

Characteristics of a Rectangular Clock Signal

T_c - clock cycle time, or period, or cycle length

$F_c = {}^1/_{T_c}$ - clock frequency



Clock frequency sets the tempo for the processor to execute instructions

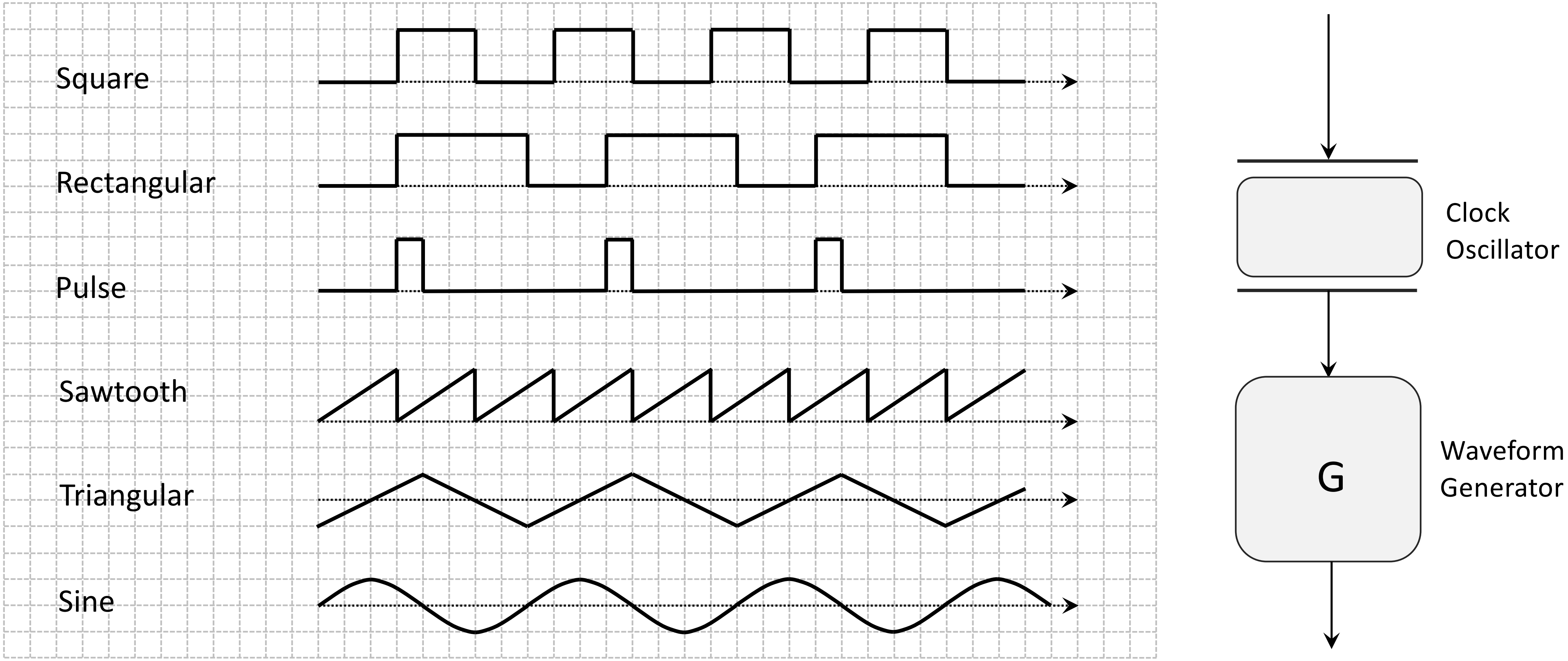
Common characteristicts for modern general-purpose CPUs:

T_c - a few nanoseconds, $\sim 10^{-9}$ sec

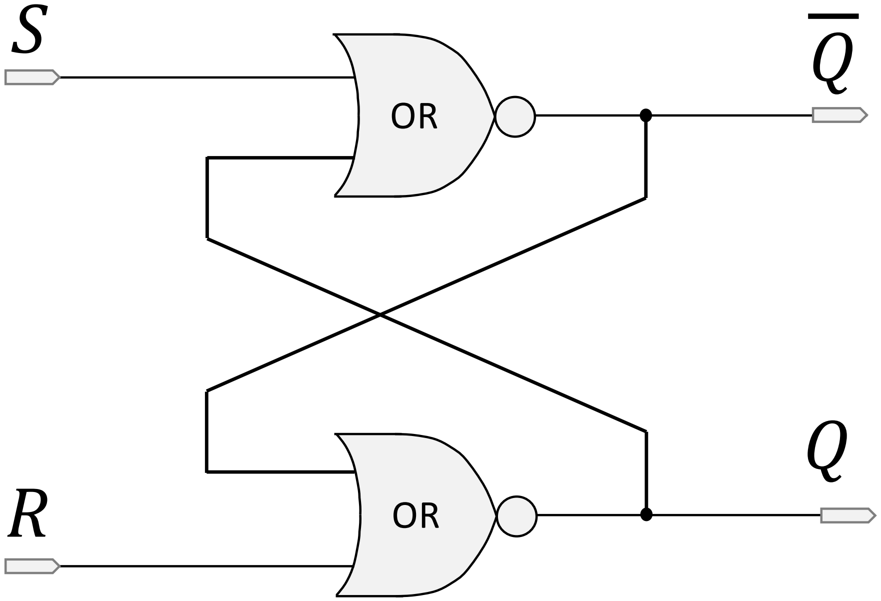
F_c - a few gigahertz, 2-3 GHz

Clock Signal Variations

(A waveform generator is used to control the form of a clock wave signal)

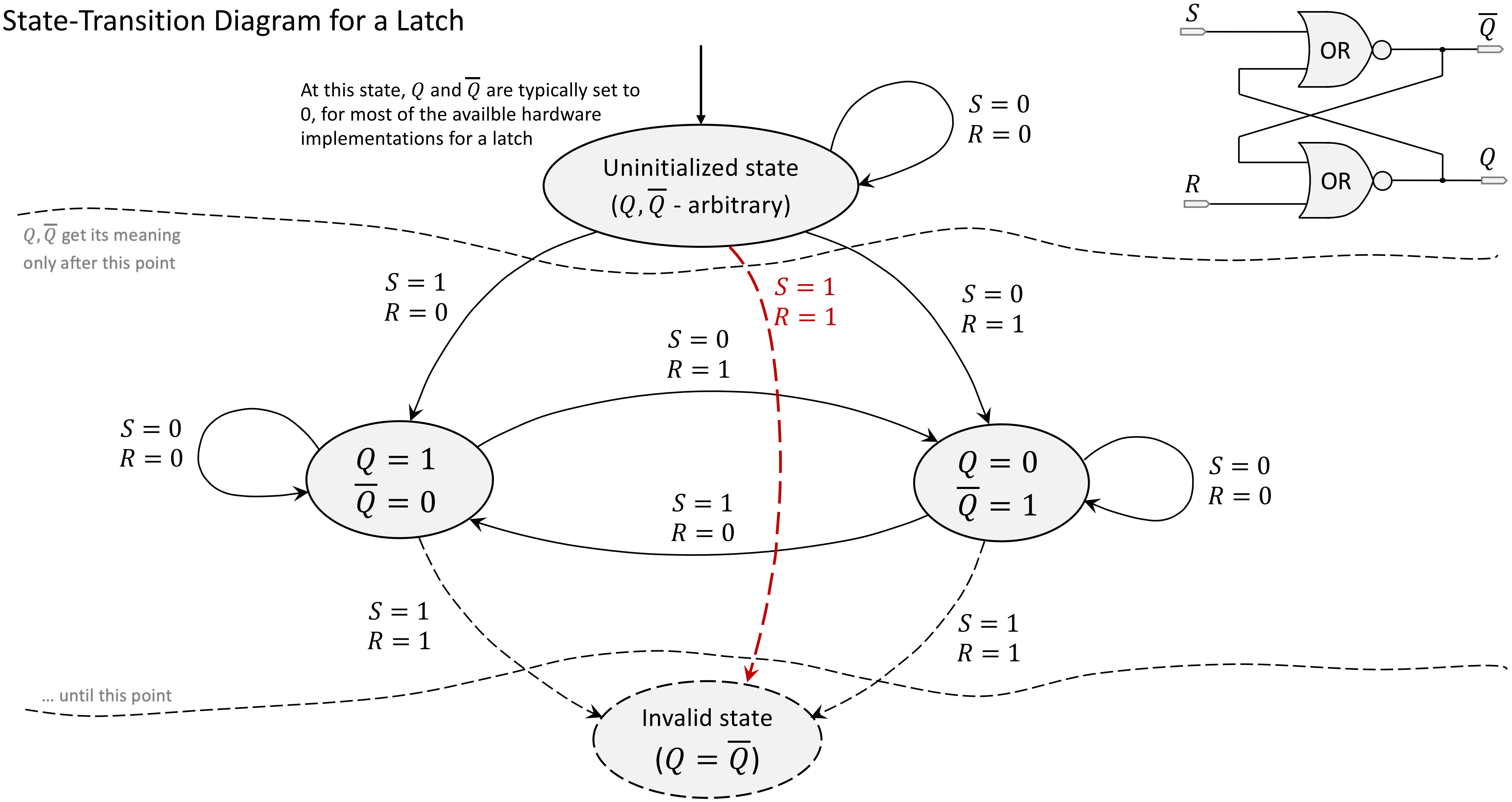


The Recap on Latch: a circuit to store one bit of information



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

State-Transition Diagram for a Latch



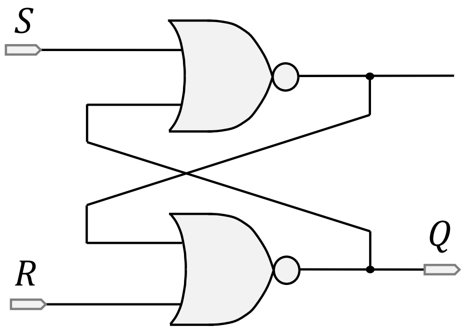
At this state, Q and $\overline{Q}$ are typically set to 0, for most of the availble hardware implementations for a latch

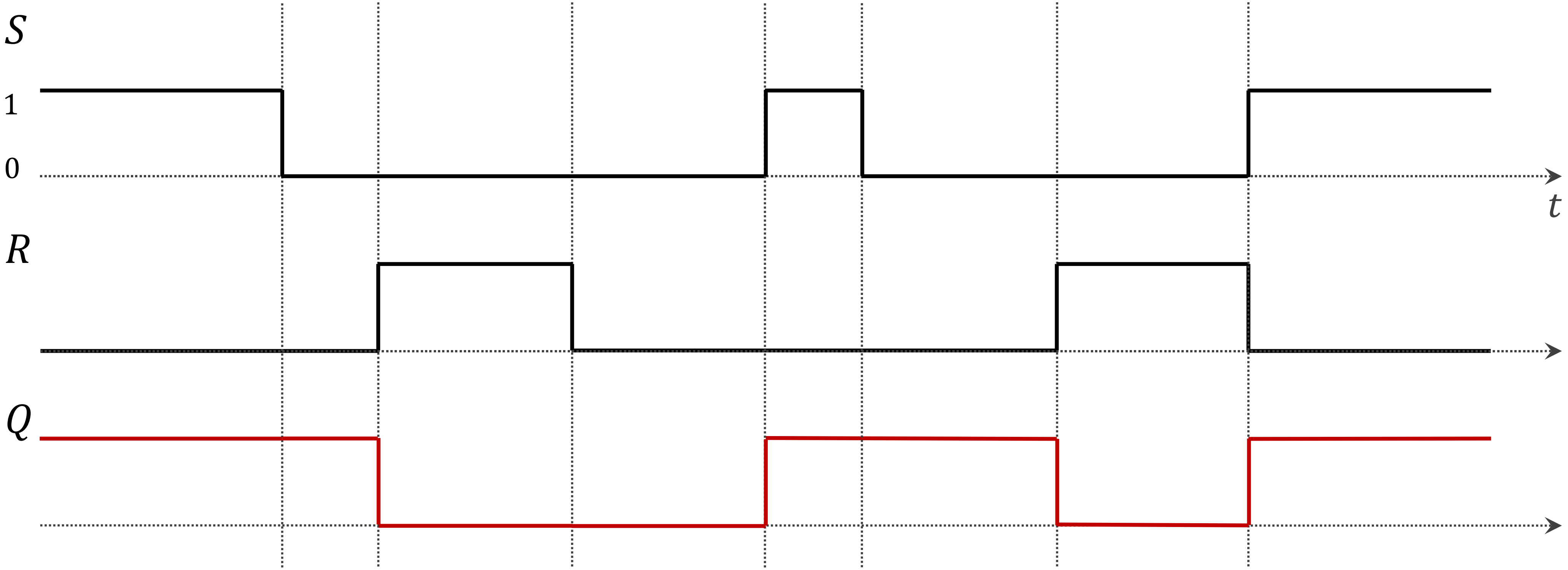
$Q, \overline{Q}$ get its meaning only after this point

… until this point

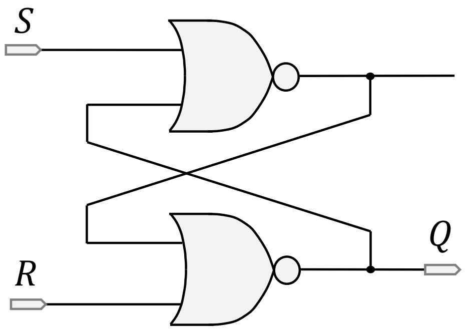
The Recap of a Transparent S/R Latch

(Transparent = the change of inputs triggers immediately the update of the output)

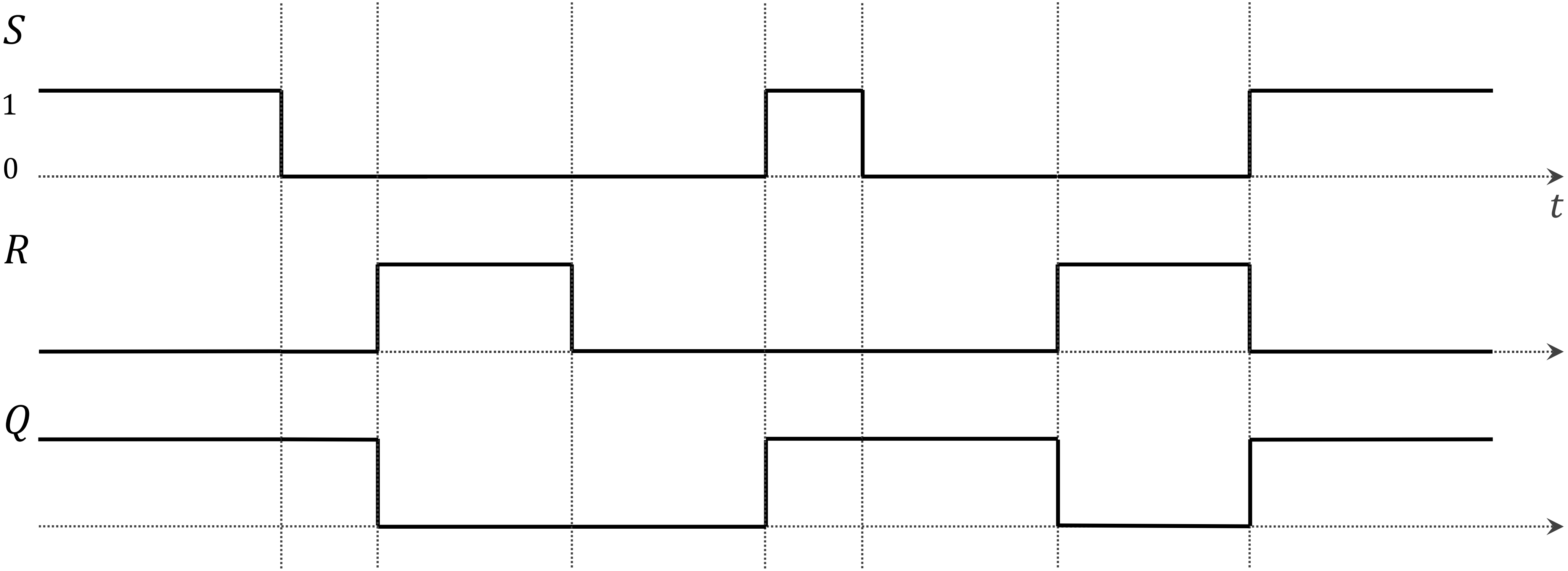




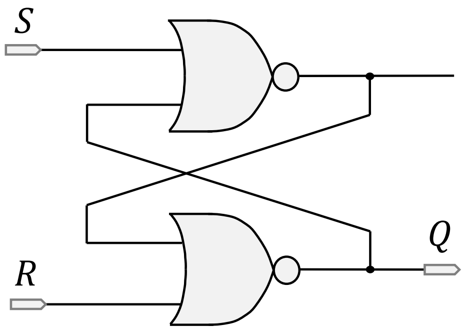
The Recap of a Transparent S/R Latch



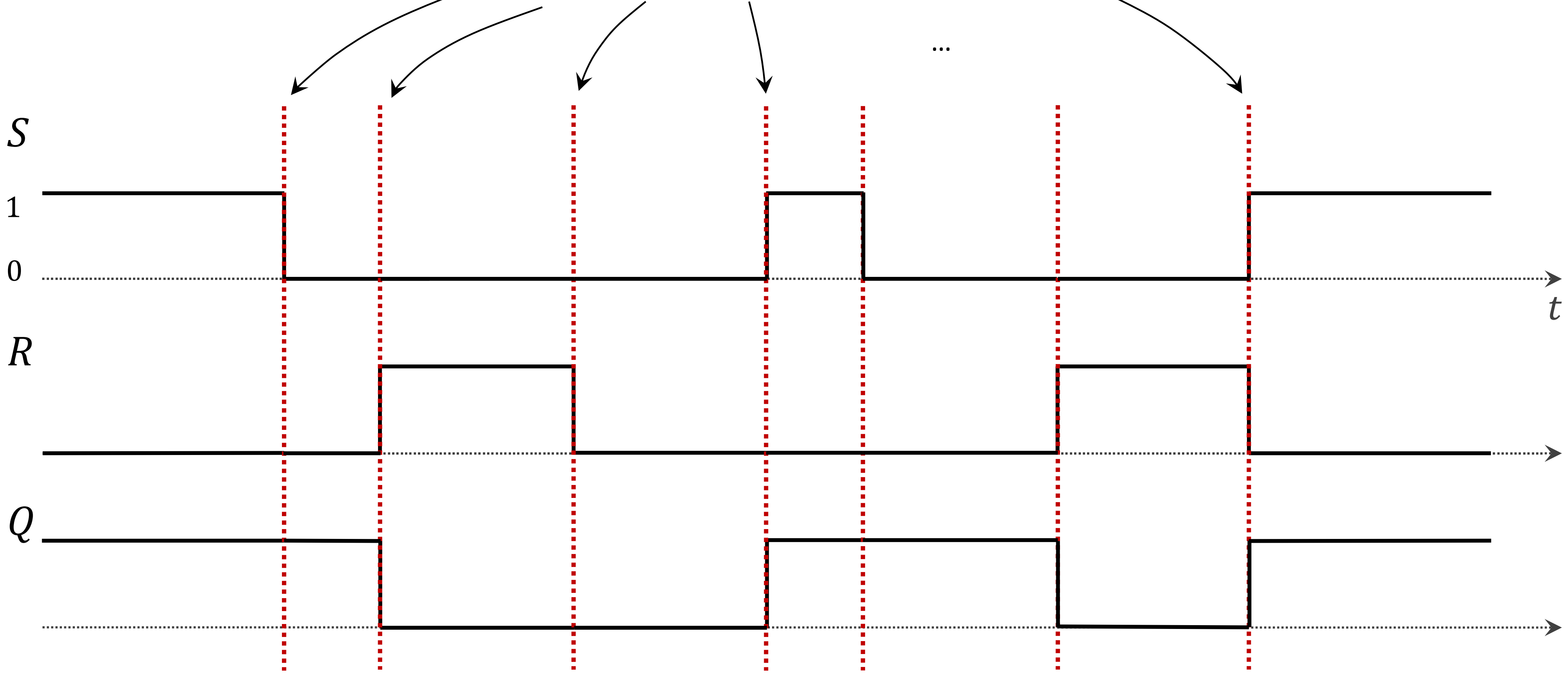
An idealistic case (no propagation overhead considered):



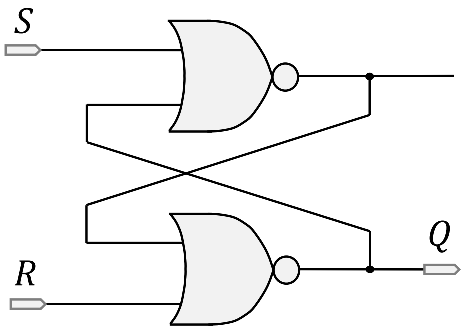
The Recap of a Transparent S/R Latch



Values for inputs S and R are produced at these times (somewhere in neighbouring subcircuits)

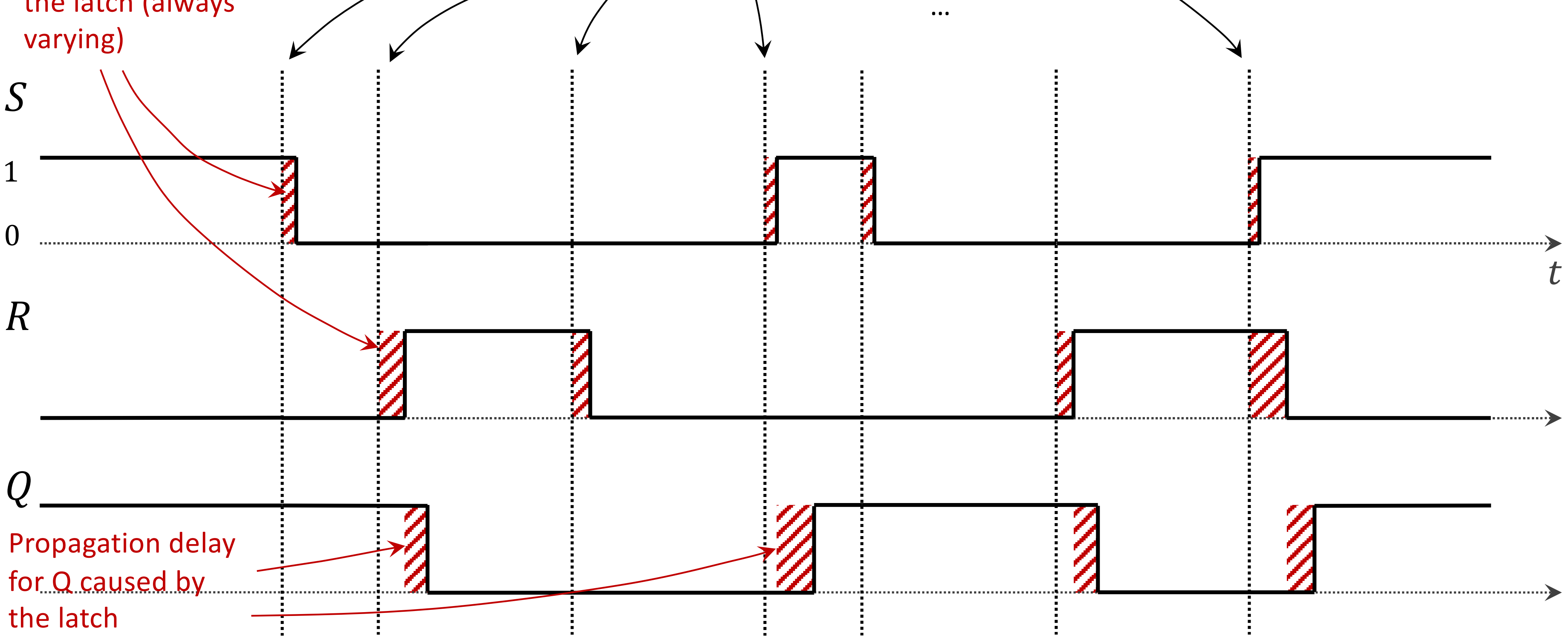


The Recap of a Transparent S/R Latch

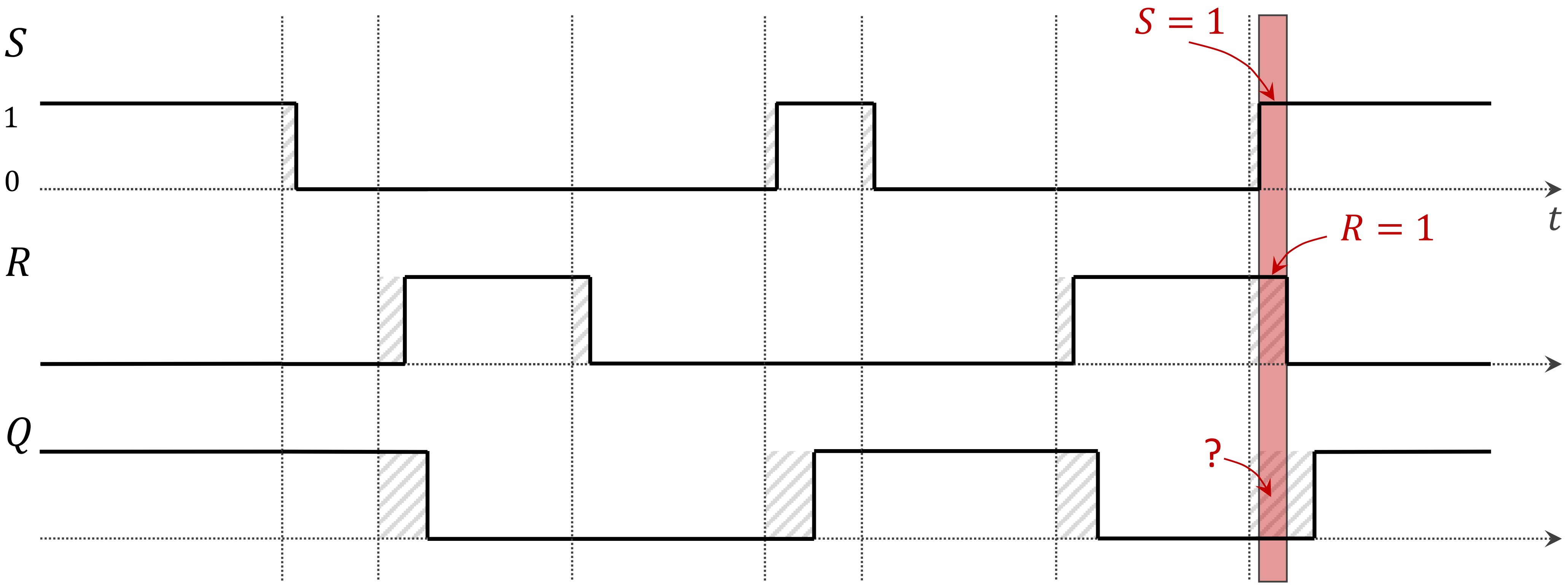


Values for inputs S and R are produced at these times (somewhere in neighbouring subcircuits)

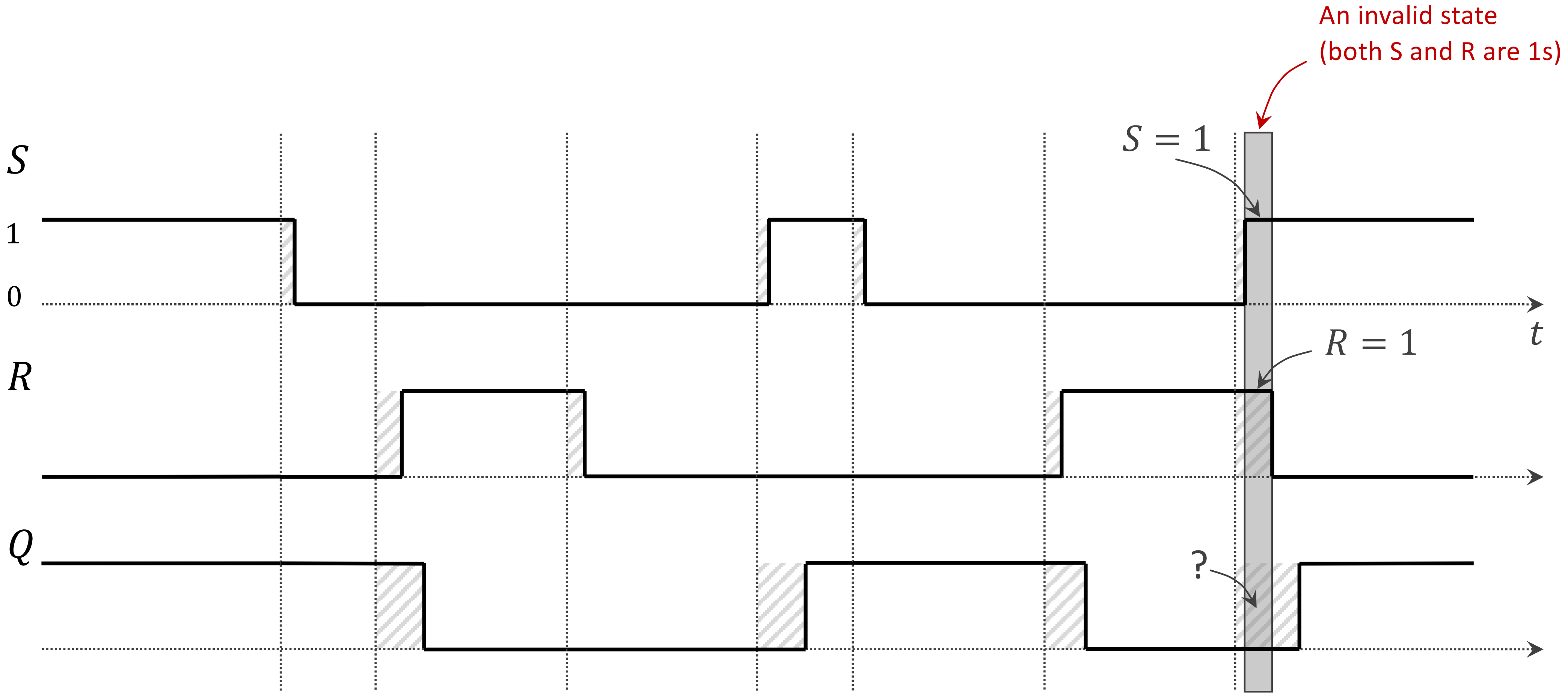
Propagation delays for S and R, to reach the latch (always varying)



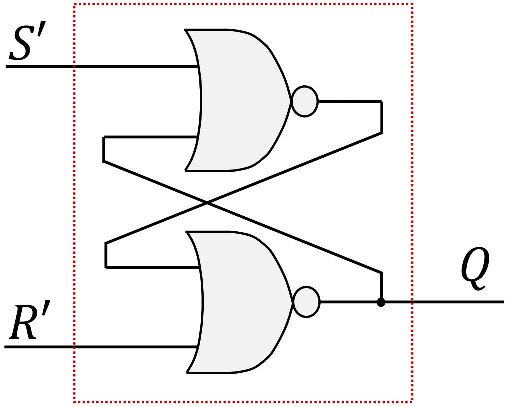
Propagation delay for Q caused by the latch


S
1
0
$S = 1$
t
R
$R = 1$
Q
?

Various problems might be caused by unpredicted propagation delays, such as an invalid state for a transparent S/R latch

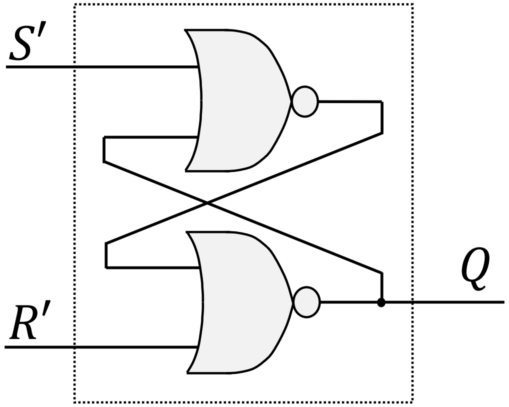


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



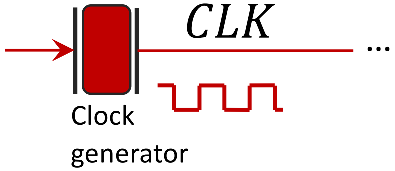
S/R Latch

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

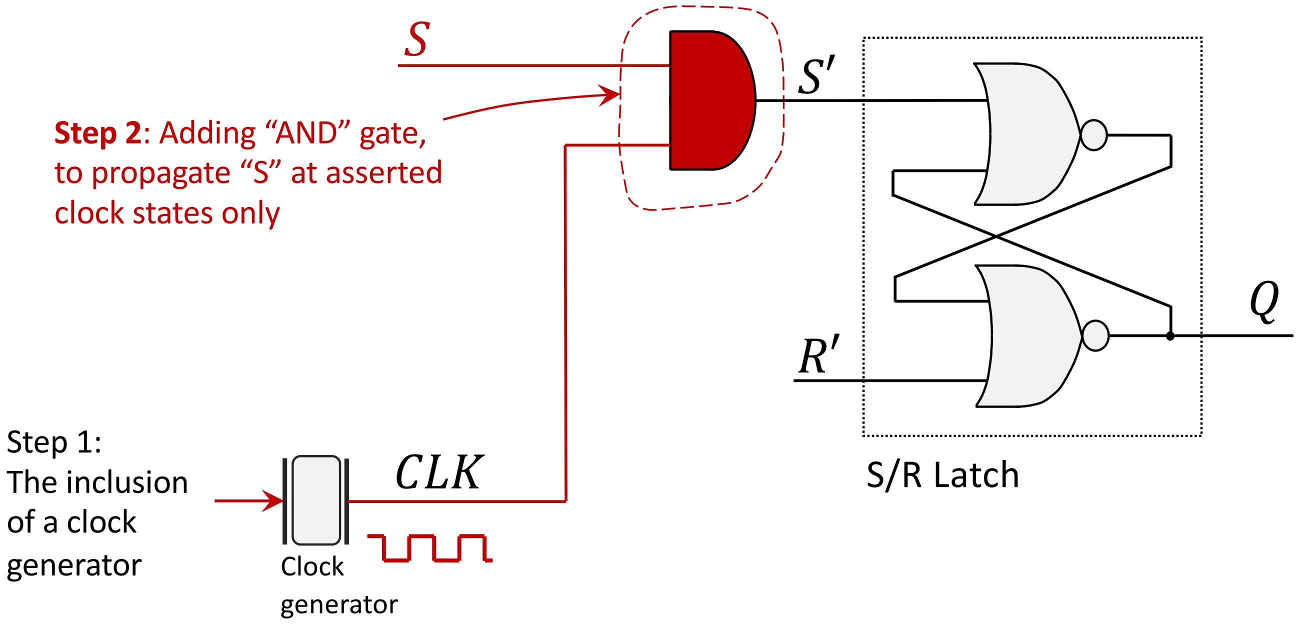


S/R Latch

Step 1:
The inclusion of a clock generator



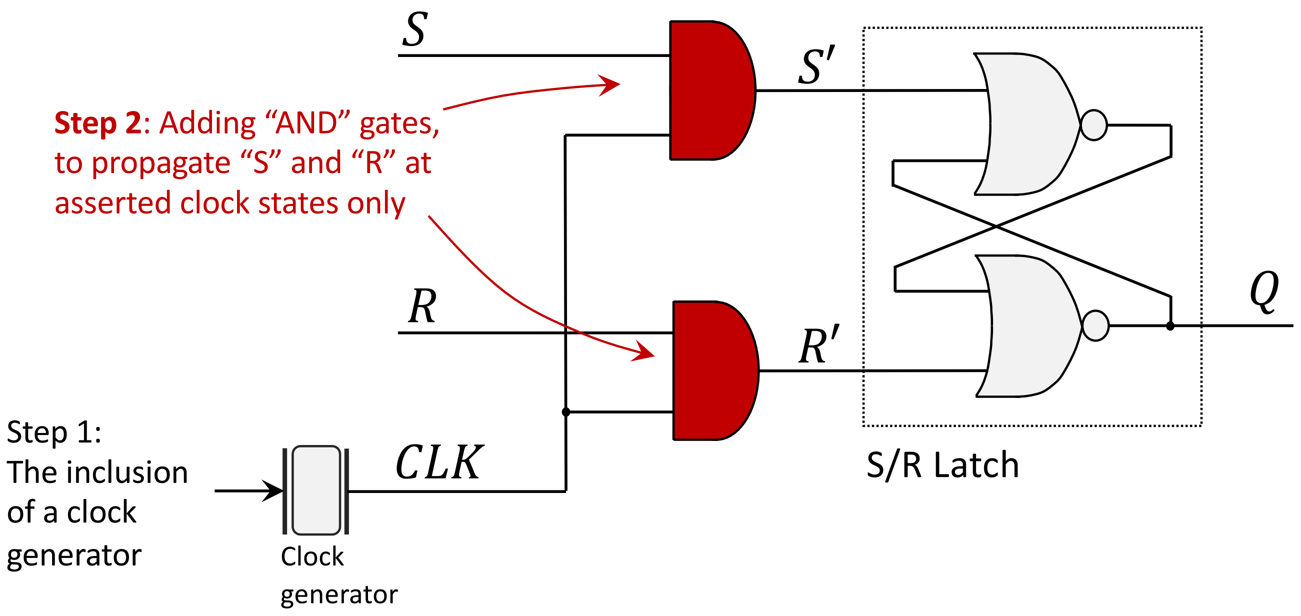
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Step 2: Adding “AND” gate, to propagate “S” at asserted clock states only

Step 1: The inclusion of a clock generator

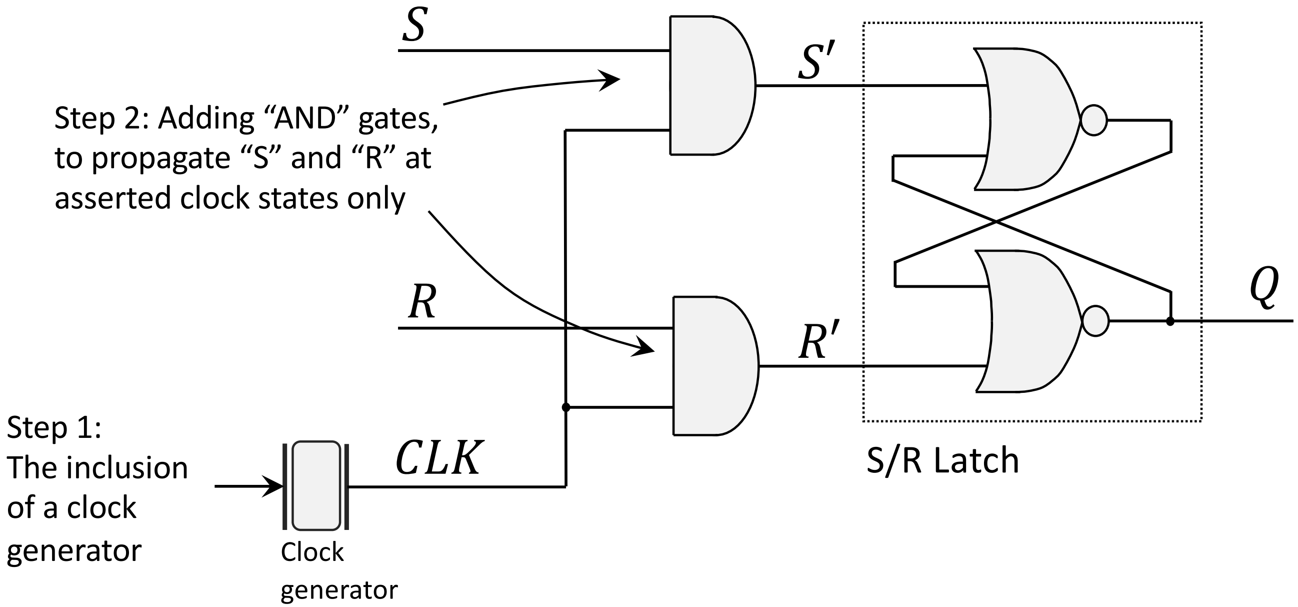
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Step 2: Adding “AND” gates, to propagate “S” and “R” at asserted clock states only

Step 1: The inclusion of a clock generator

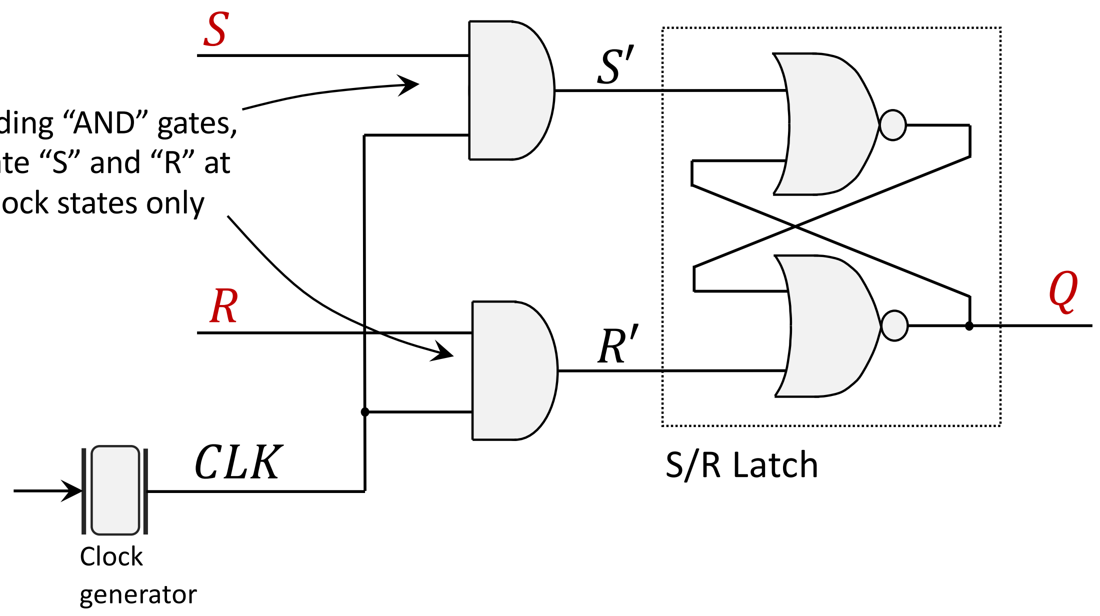
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Step 2: Adding “AND” gates, to propagate “S” and “R” at asserted clock states only

Step 1: The inclusion of a clock generator

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



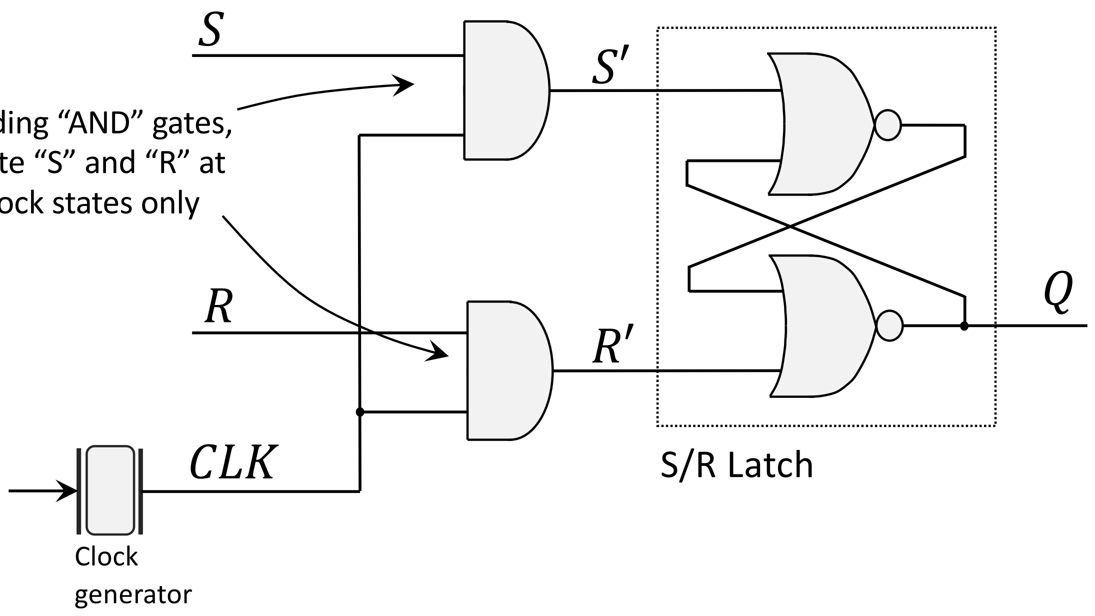
Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 1: The inclusion of a clock generator

Observation:

Inputs "S" and "R" make effect on "Q" only when clock is asserted

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

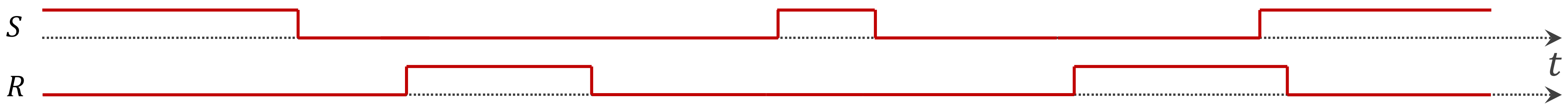


Step 2: Adding “AND” gates, to propagate “S” and “R” at asserted clock states only

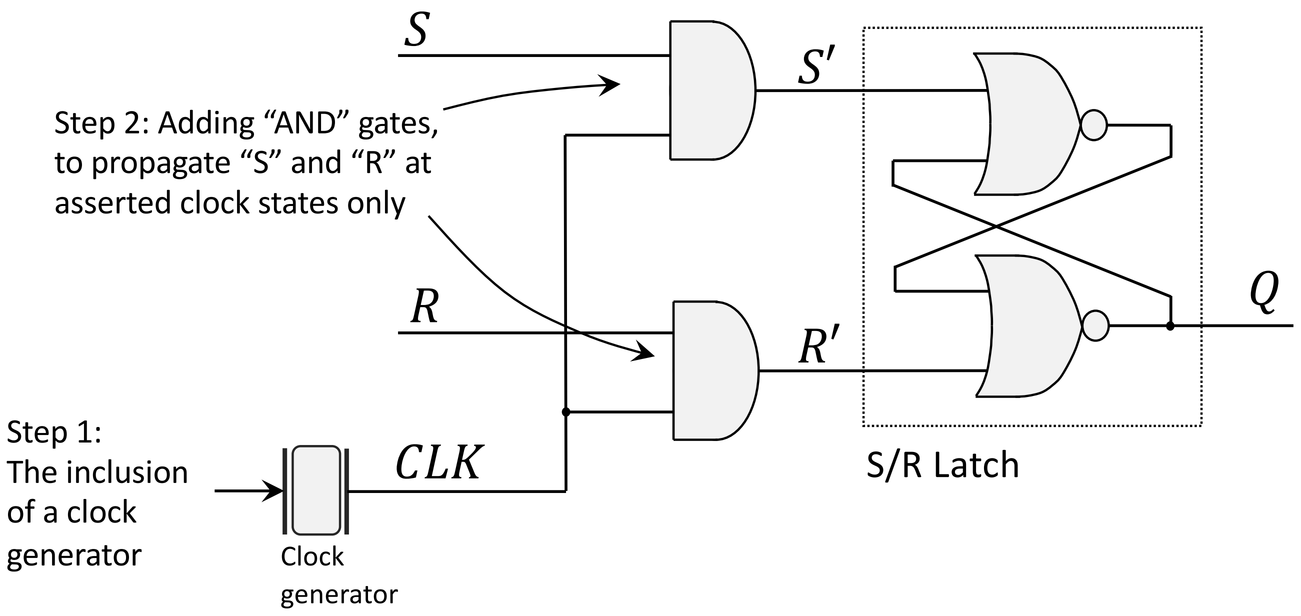
Step 1: The inclusion of a clock generator

Observation:

Inputs “S” and “R” make effect on “Q” only when clock is asserted



Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

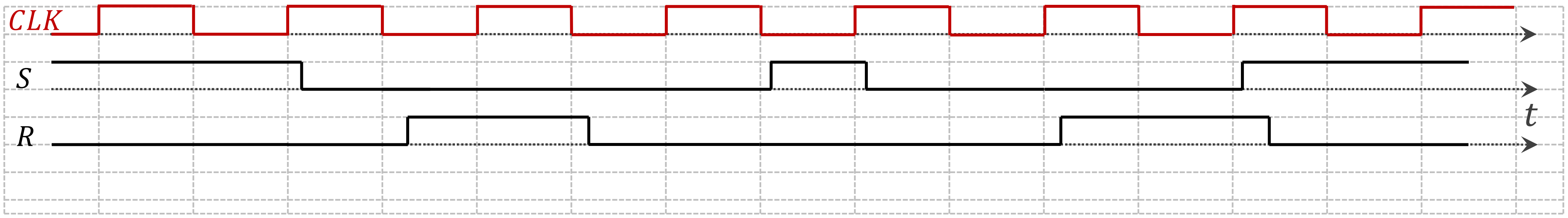


Step 2: Adding “AND” gates, to propagate “S” and “R” at asserted clock states only

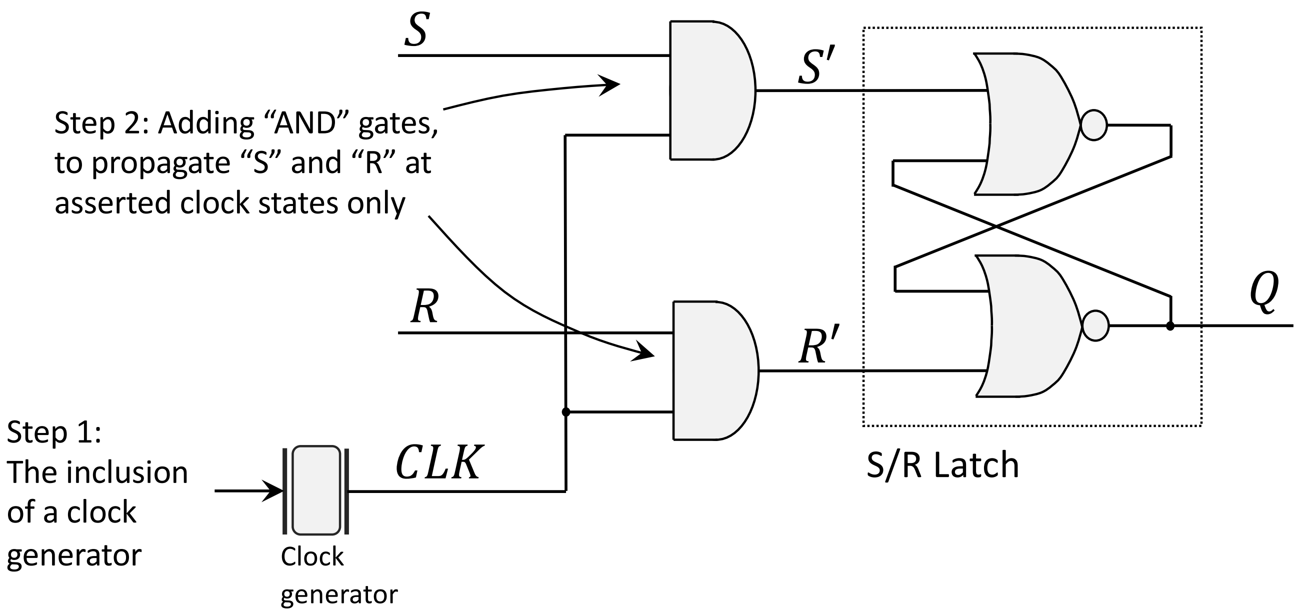
Step 1: The inclusion of a clock generator

Observation:

Inputs “S” and “R” make effect on “Q” only when clock is asserted



Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

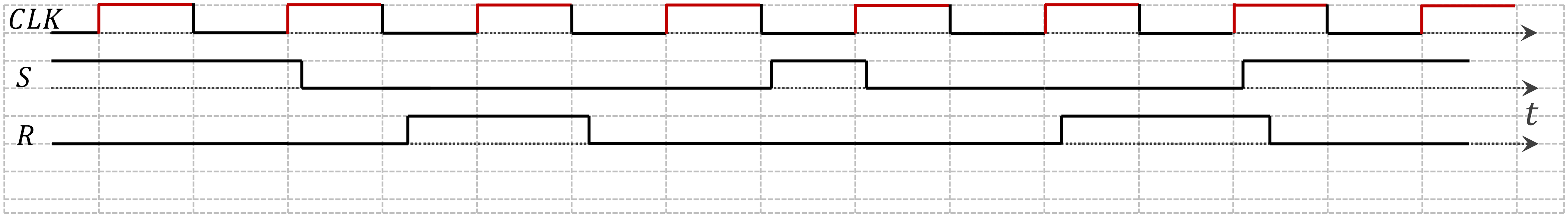


Step 2: Adding “AND” gates, to propagate “S” and “R” at asserted clock states only

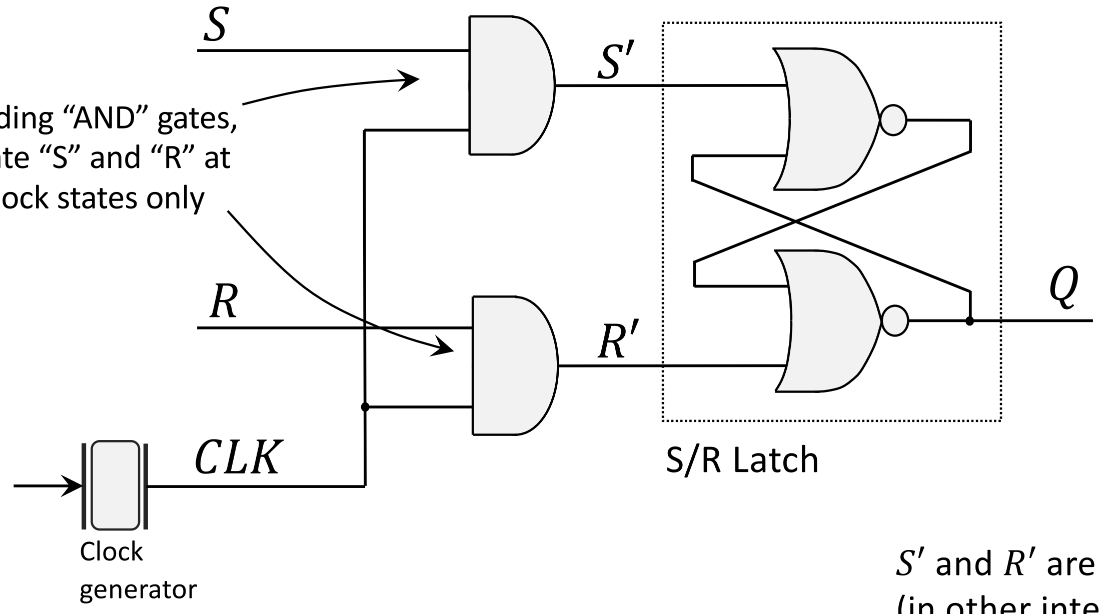
Step 1:
The inclusion of a clock generator

Observation:

Inputs “S” and “R” make effect on “Q” only when clock is asserted



Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



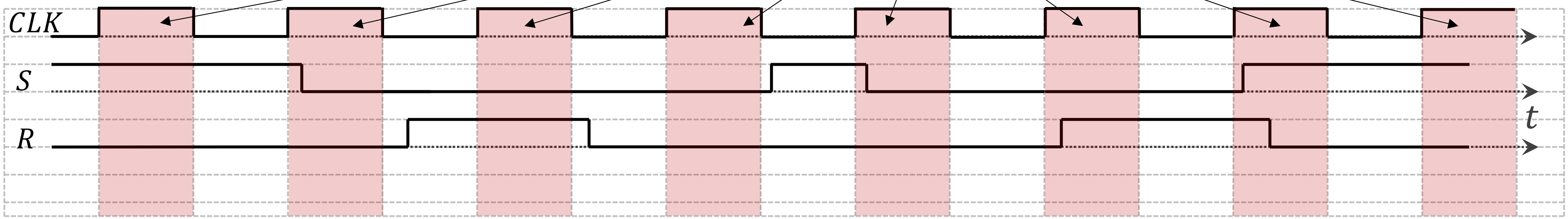
Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 1: The inclusion of a clock generator

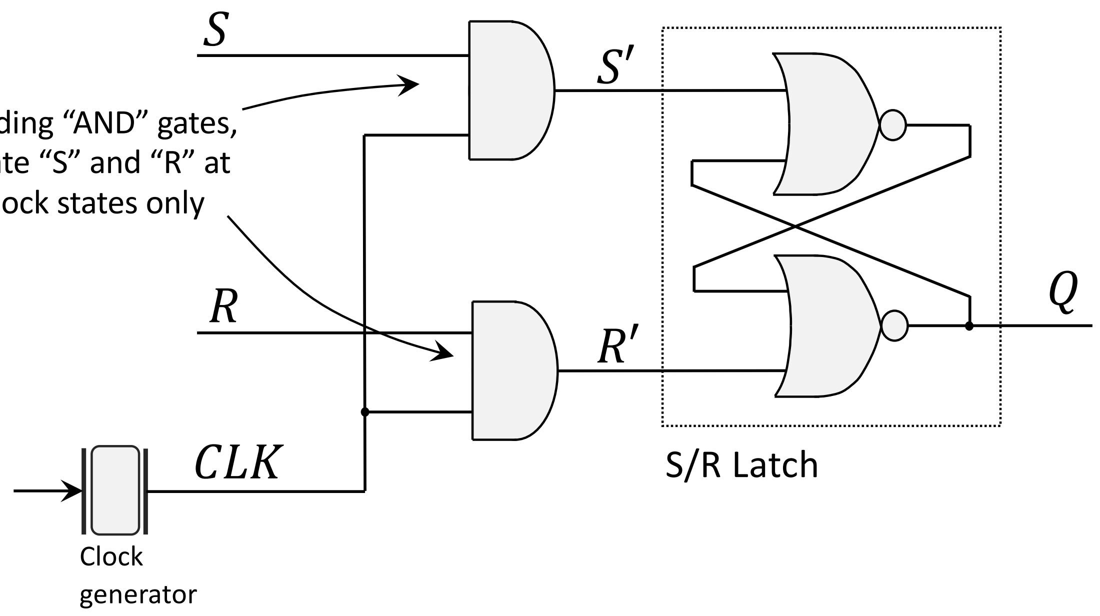
Observation:

Inputs "S" and "R" make effect on "Q" only when clock is asserted

S' and R' are updated only at these time intervals (in other intervals they equal to 0s)

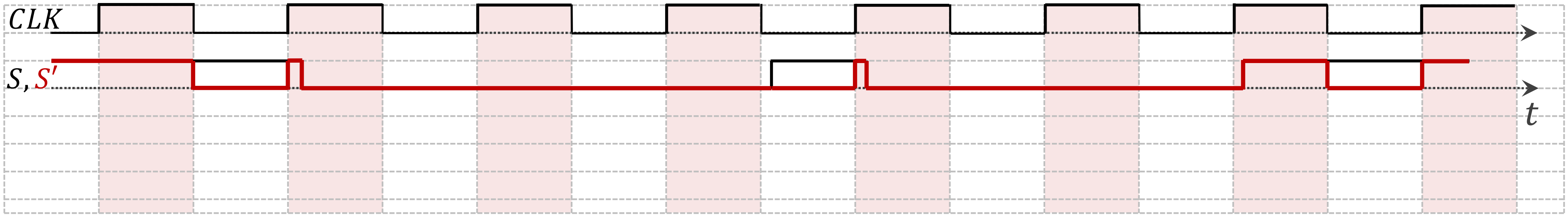


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

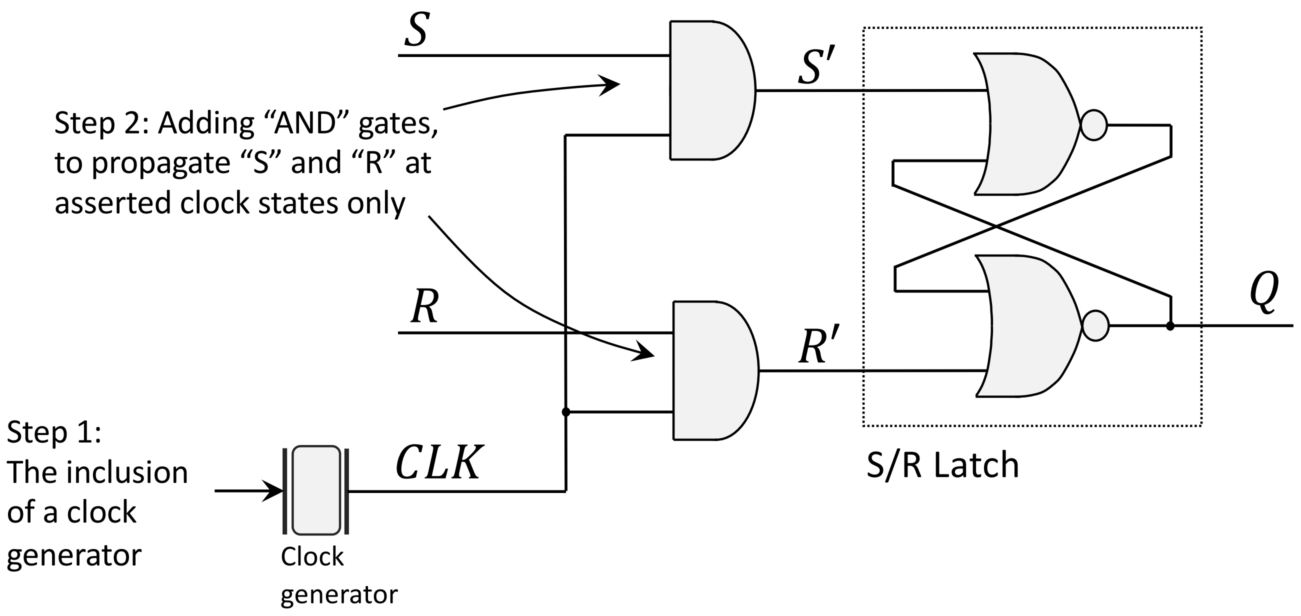


Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 1:
The inclusion of a clock generator

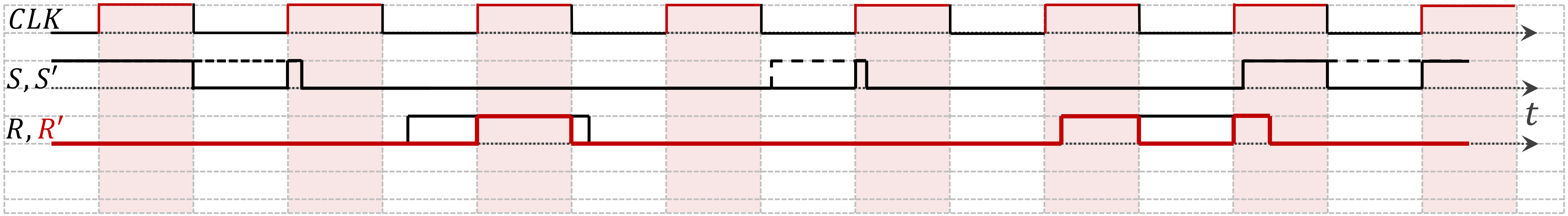


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

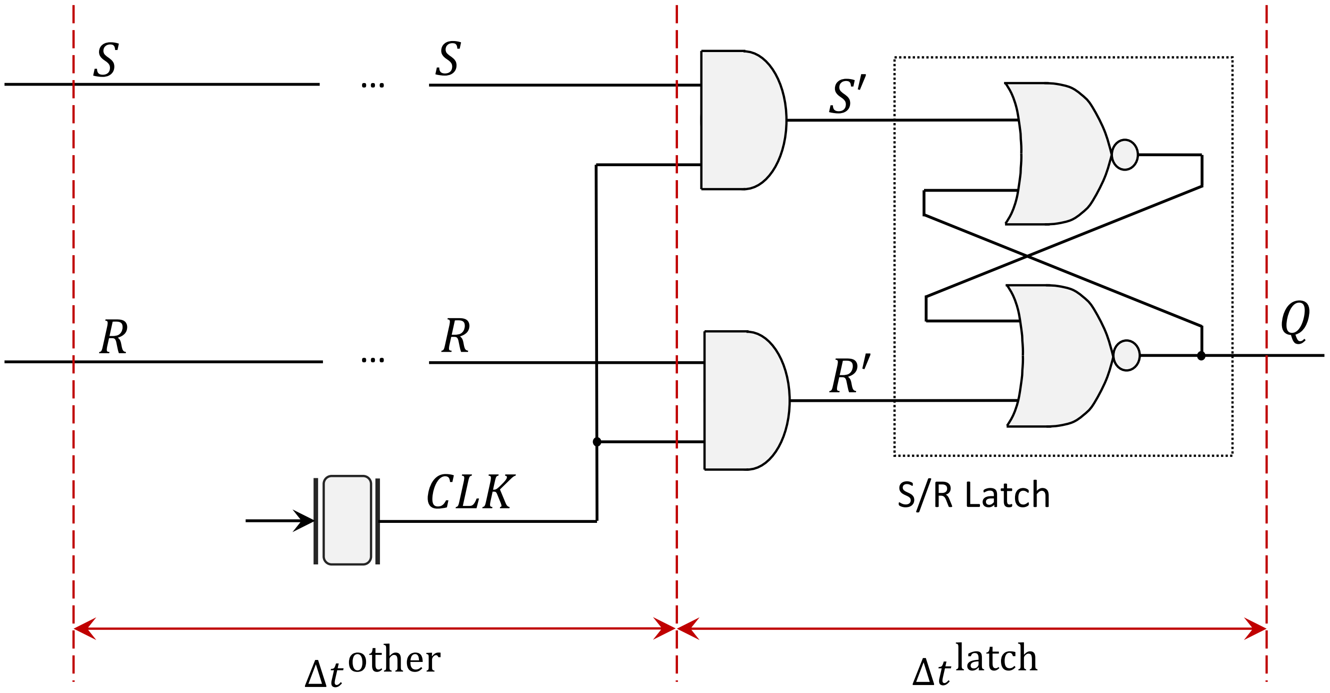


Step 2: Adding “AND” gates, to propagate “S” and “R” at asserted clock states only

Step 1: The inclusion of a clock generator

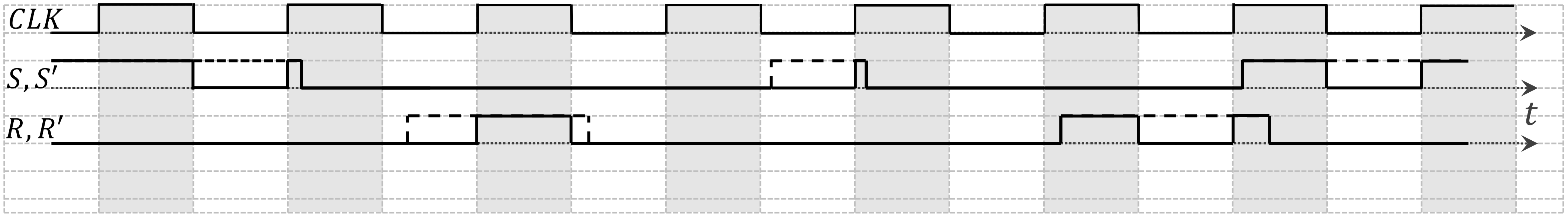


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

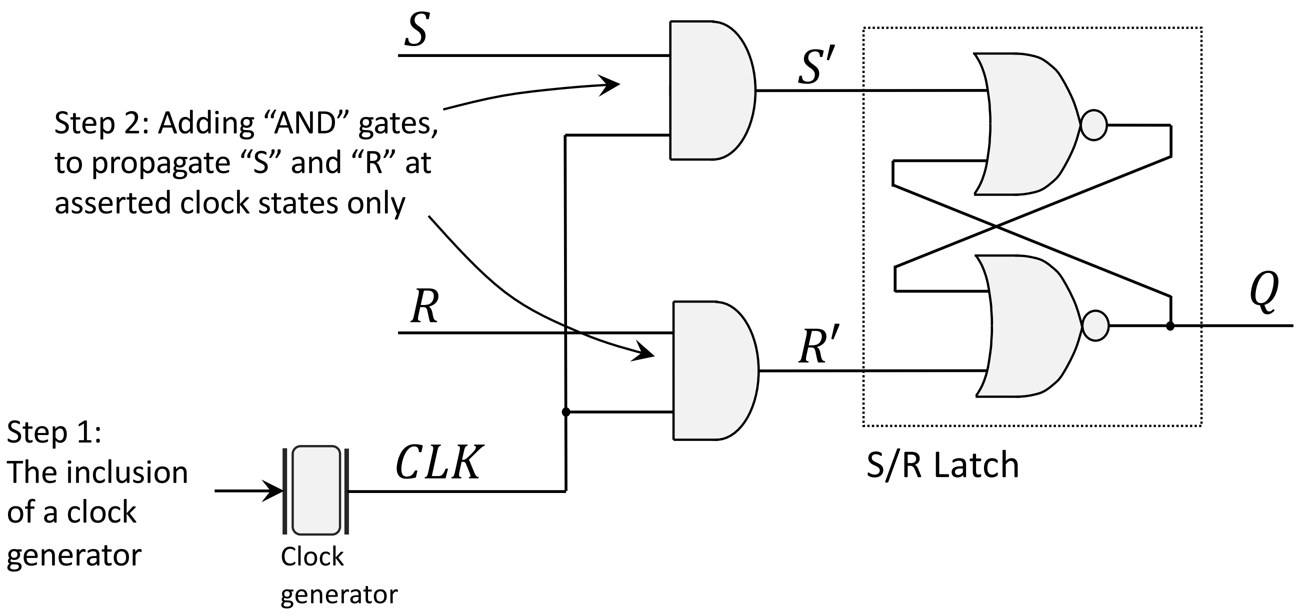


Assumption:

Latch propagation delay Δt^{latch} is insignificant, as compared to Δt^{other}

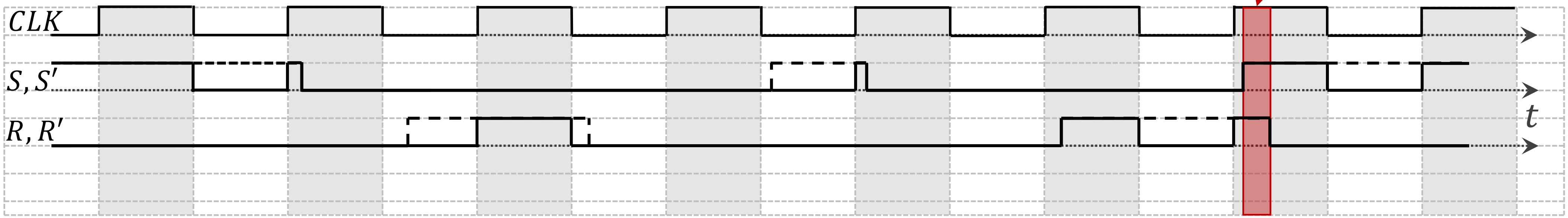


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



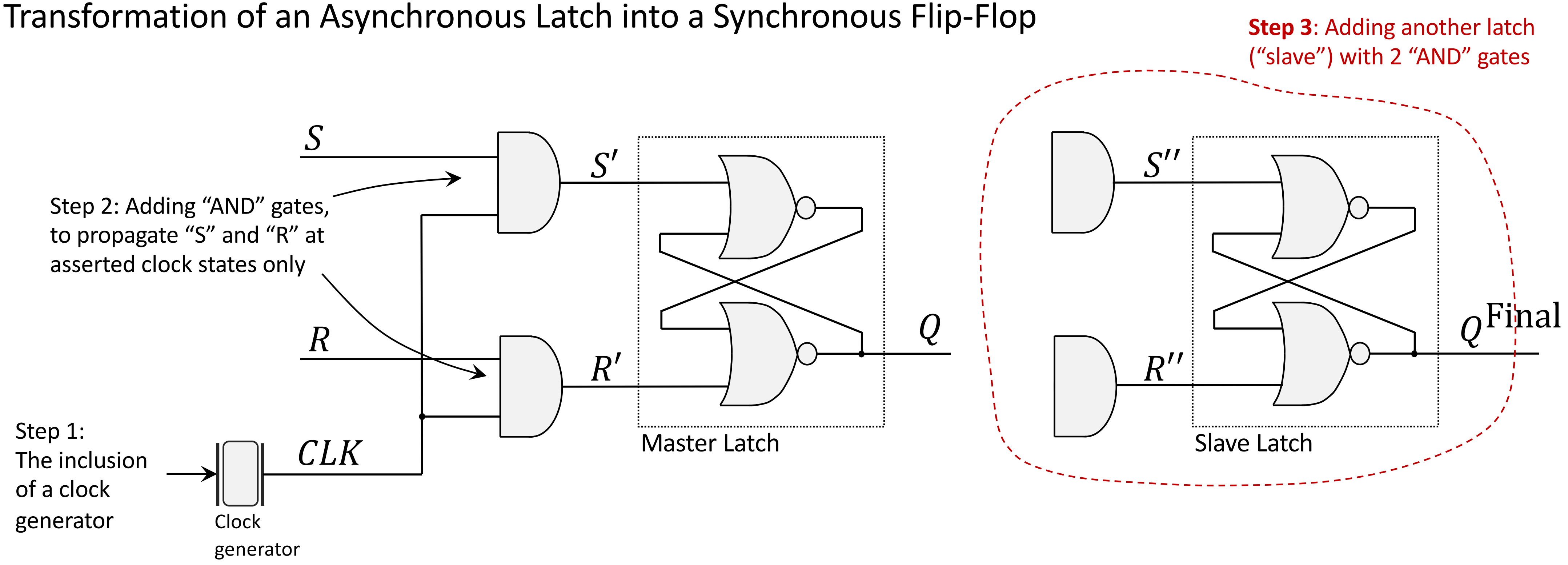
Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 1: The inclusion of a clock generator



An invalid state (both S and R are 1s) remains

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

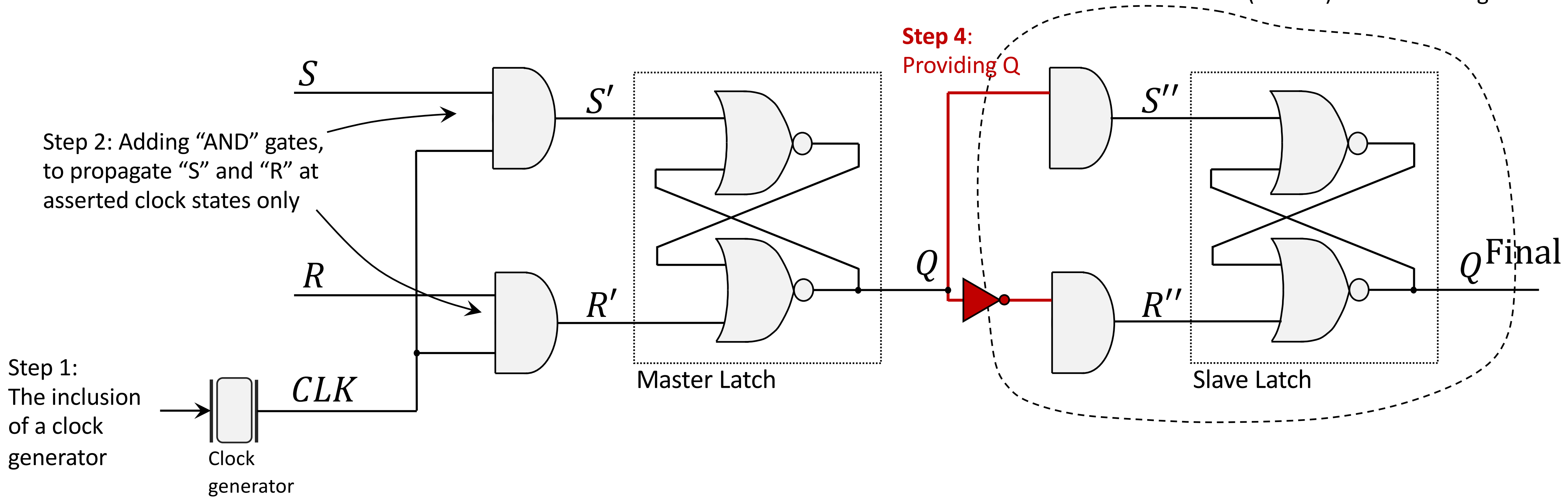


Step 3: Adding another latch ("slave") with 2 "AND" gates

Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 1: The inclusion of a clock generator

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



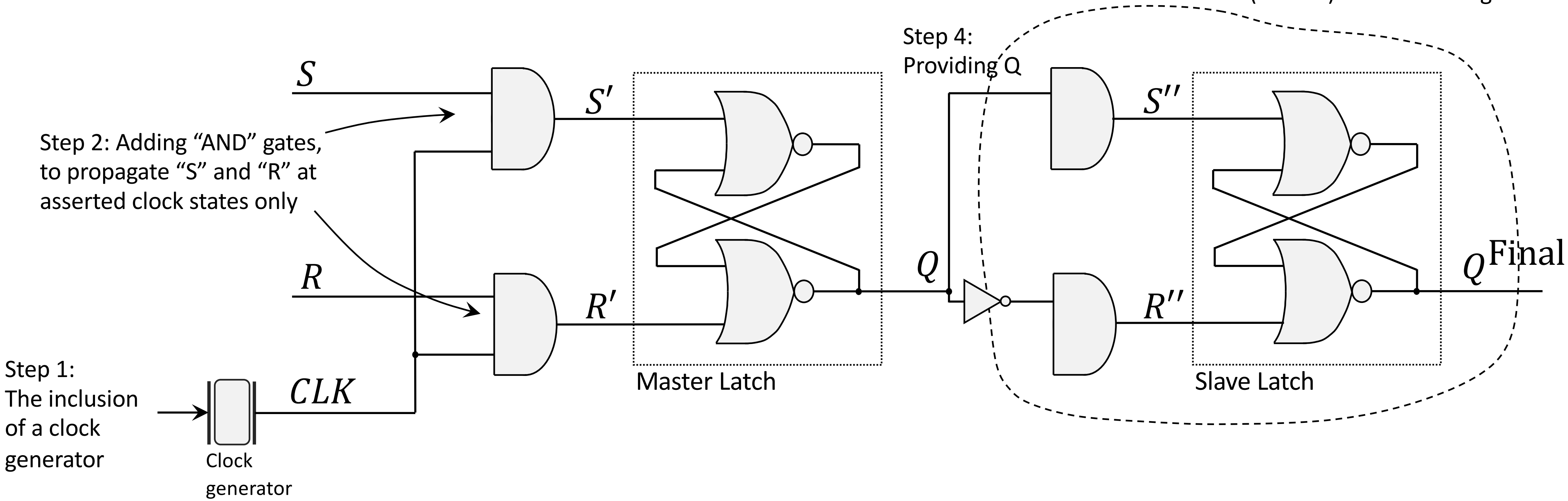
Step 1: The inclusion of a clock generator

Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 3: Adding another latch ("slave") with 2 "AND" gates

Step 4: Providing Q

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Step 3: Adding another latch ("slave") with 2 "AND" gates

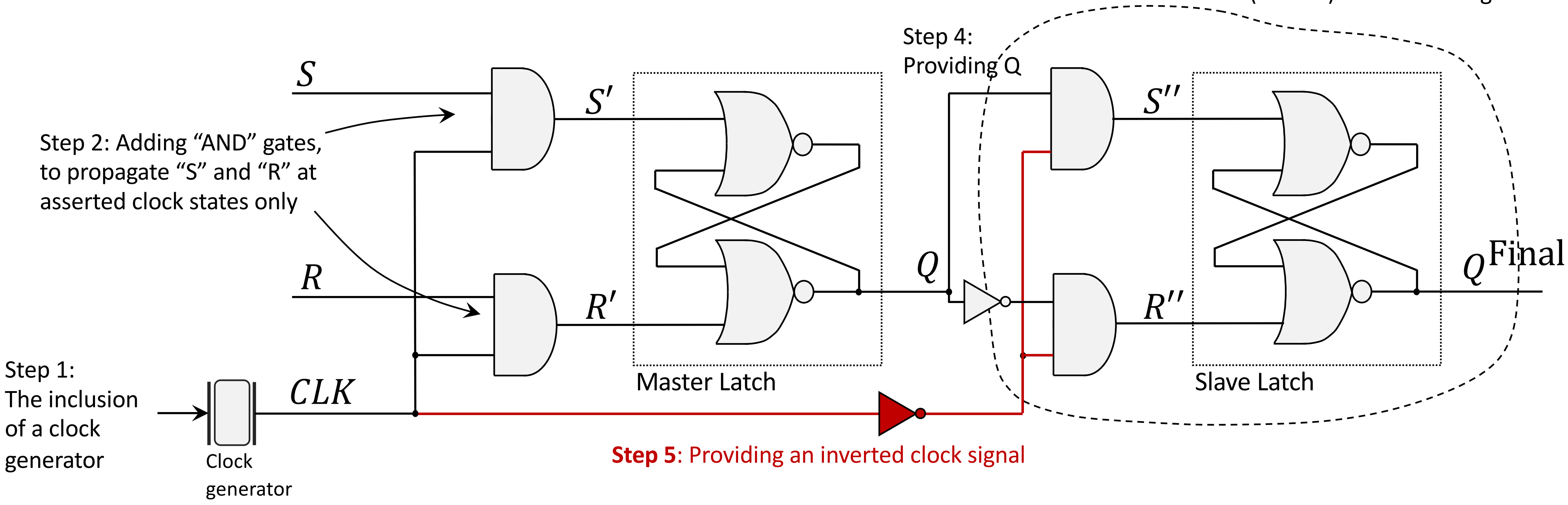
Step 4: Providing Q

Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 1: The inclusion of a clock generator

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

Step 3: Adding another latch ("slave") with 2 "AND" gates



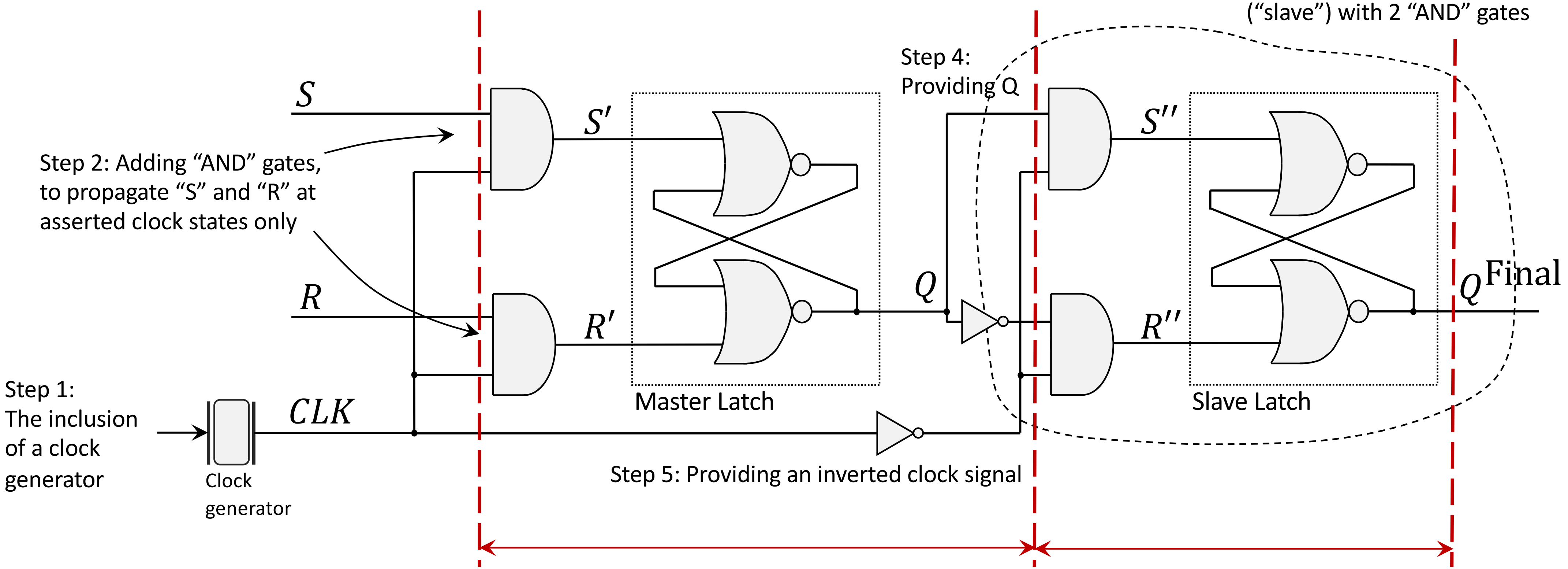
Step 4: Providing Q

Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 1: The inclusion of a clock generator

Step 5: Providing an inverted clock signal

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Step 1: The inclusion of a clock generator

Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 3: Adding another latch ("slave") with 2 "AND" gates

Step 4: Providing Q

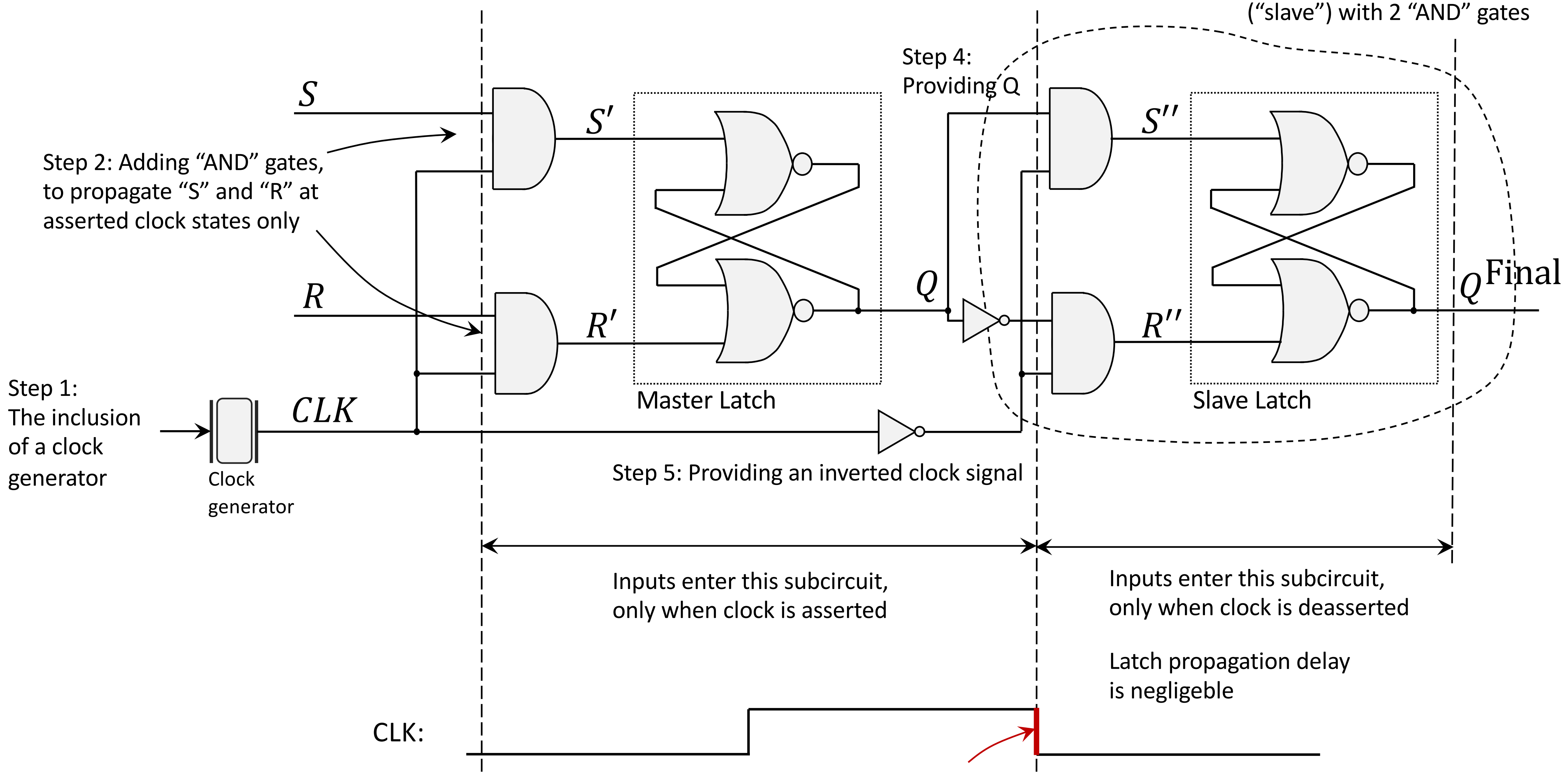
Step 5: Providing an inverted clock signal

Inputs enter this subcircuit, only when clock is asserted

Inputs enter this subcircuit, only when clock is deasserted

Latch propagation delay is negligeble

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Step 3: Adding another latch ("slave") with 2 "AND" gates

Step 4: Providing Q

Step 2: Adding "AND" gates, to propagate "S" and "R" at asserted clock states only

Step 1: The inclusion of a clock generator

Step 5: Providing an inverted clock signal

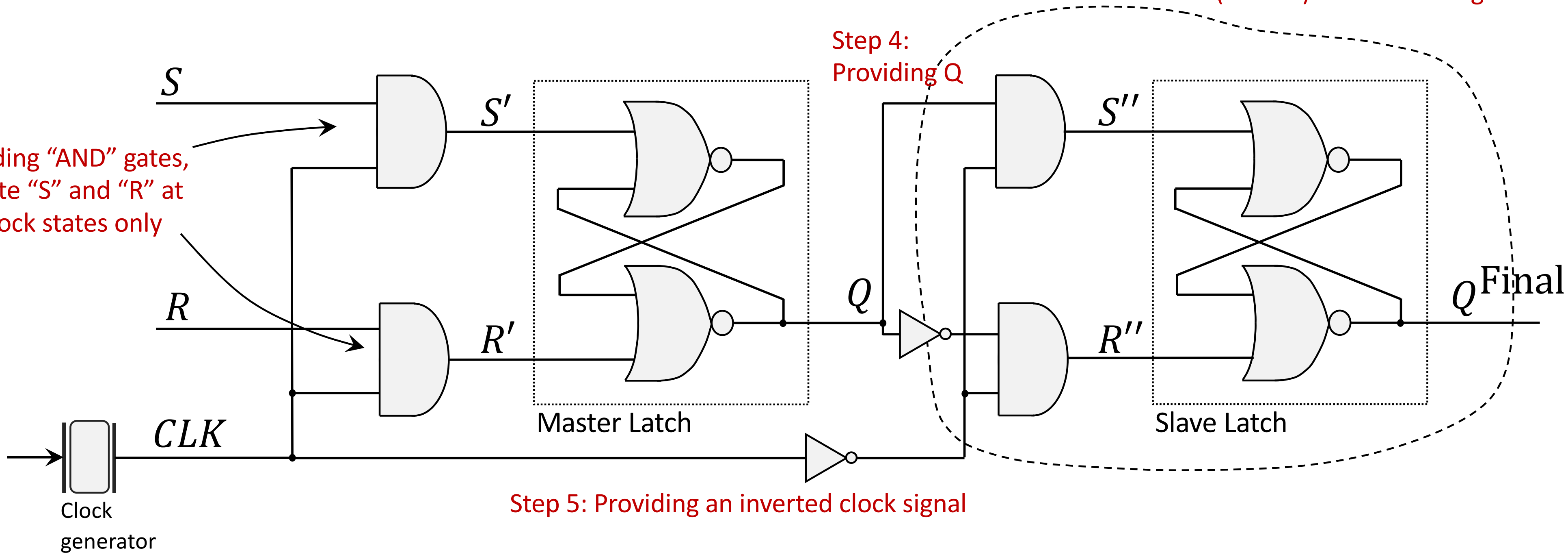
Inputs enter this subcircuit, only when clock is asserted

Inputs enter this subcircuit, only when clock is deasserted

Latch propagation delay is negligeble

Q^{Final} is updated only at the decreasing clock edge

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Step 1: The inclusion of a clock generator

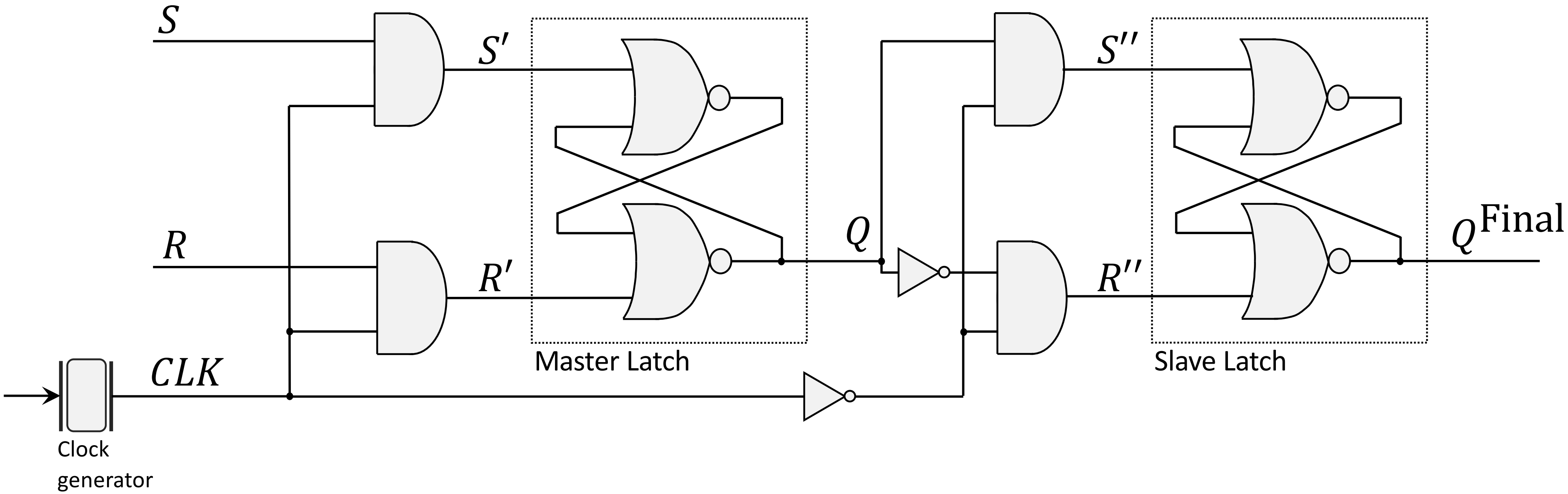
Step 2: Adding “AND” gates, to propagate “S” and “R” at asserted clock states only

Step 3: Adding another latch (“slave”) with 2 “AND” gates

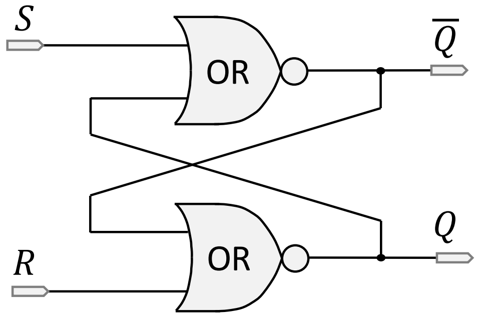
Step 4: Providing Q

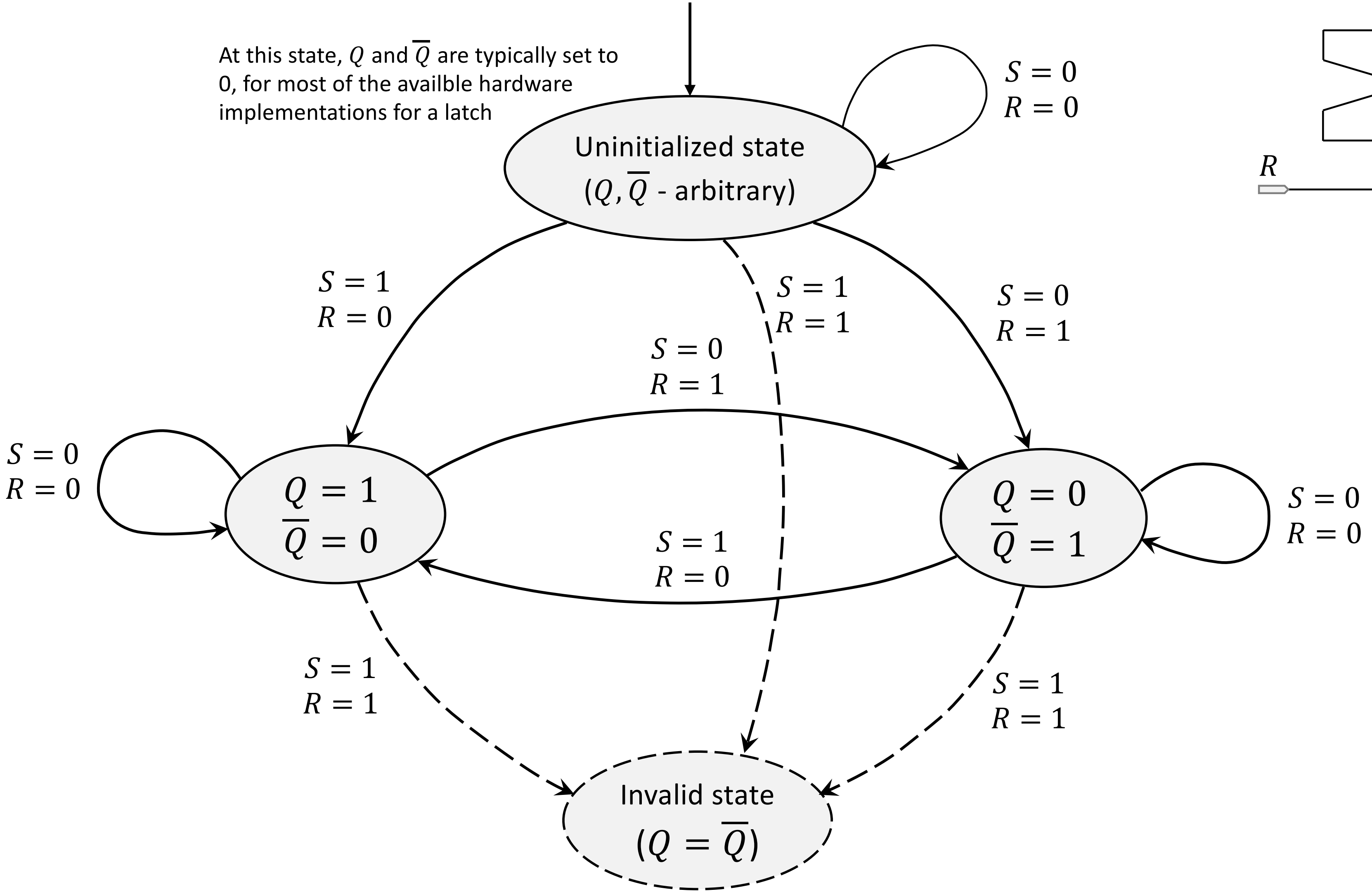
Step 5: Providing an inverted clock signal

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



State-Transition Diagram for a Latch

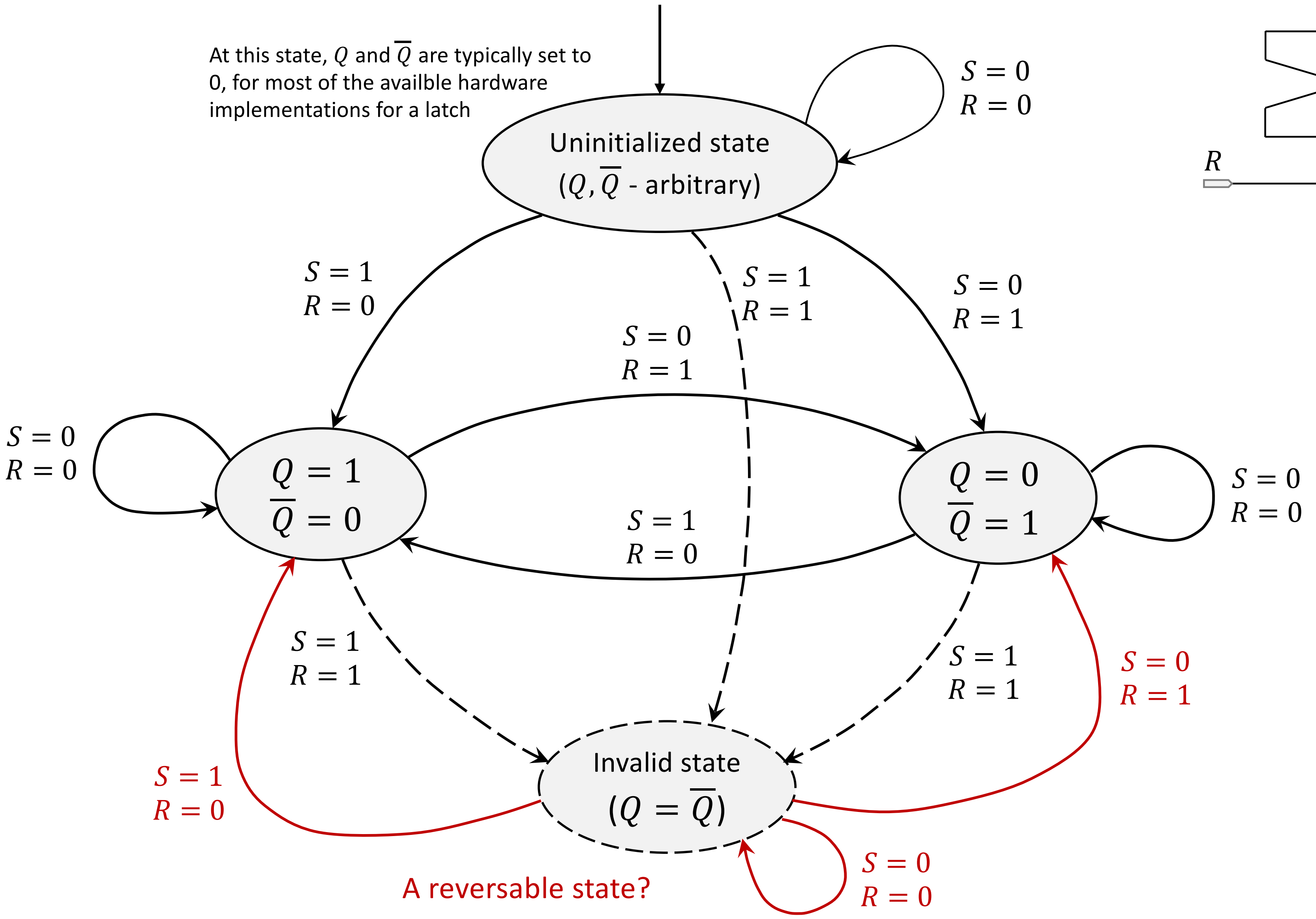




At this state, Q and $\overline{Q}$ are typically set to 0, for most of the availble hardware implementations for a latch

A reversable state?

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 for a latch

Uninitialized state

($Q, \overline{Q}$ - arbitrary)

$S = 0$ $R = 0$

$S = 1$ $R = 0$

$S = 1$ $R = 1$

$S = 0$ $R = 1$

$S = 0$ $R = 1$

$S = 0$ $R = 0$

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

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

$S = 0$ $R = 0$

$S = 1$ $R = 0$

$S = 1$ $R = 1$

$S = 1$ $R = 1$

$S = 0$ $R = 1$

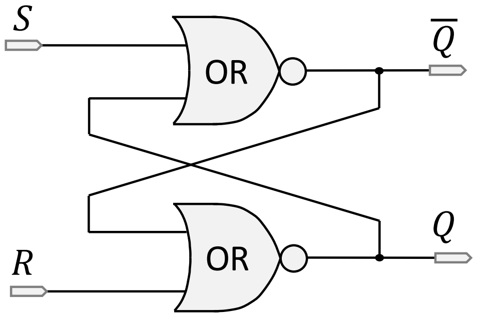
$S = 1$ $R = 0$

Invalid state

$(Q = \overline{Q})$

$S = 0$ $R = 0$

A reversable state?



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 behave 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		

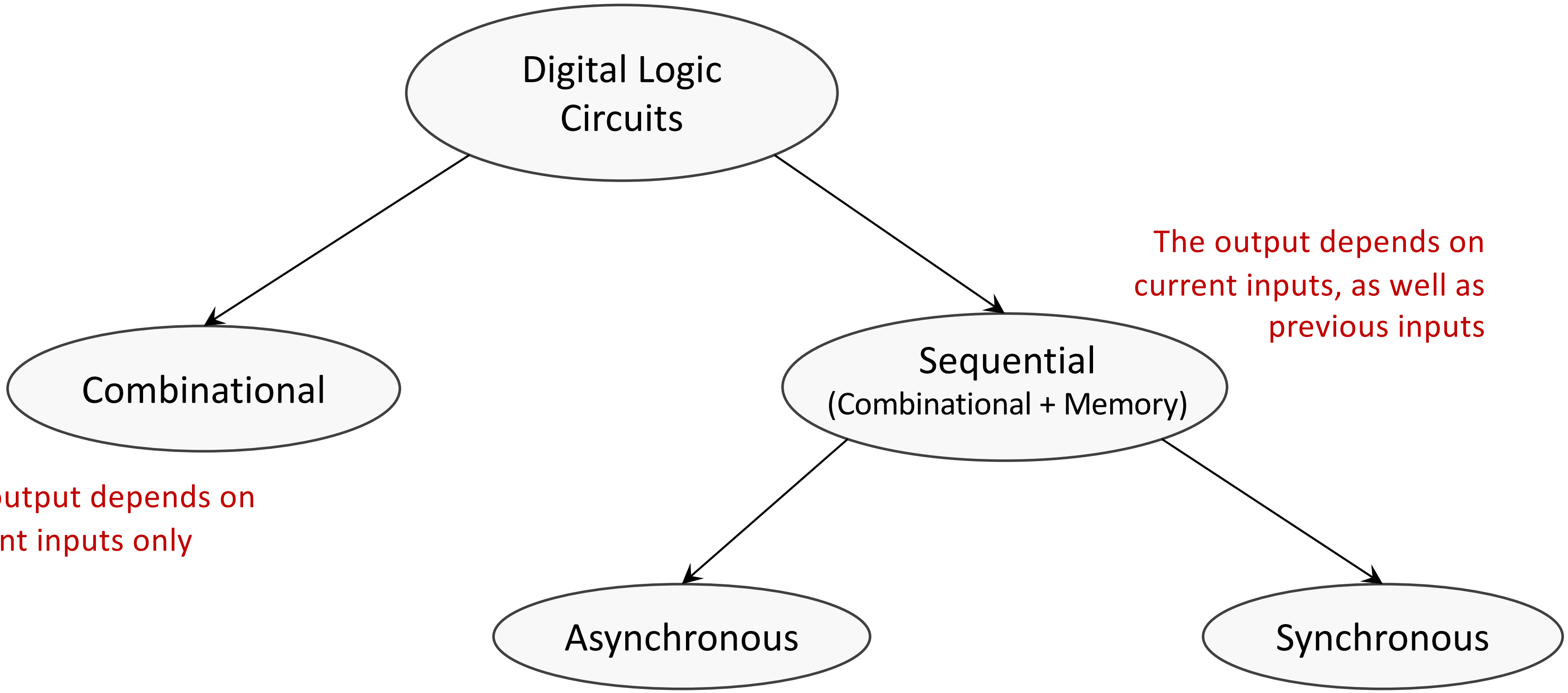
Synchronous vs. Asynchronous Circuits

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



Digital Logic Circuits

Combinational

The output depends on current inputs only

ALU

Sequential (Combinational + Memory)

The output depends on current inputs, as well as previous inputs

Asynchronous

No clock signal is used

Simplified microprocessors (Possibly, ALU as well);
Asynchronous carry-lookahead adders;
Ripple carry adders

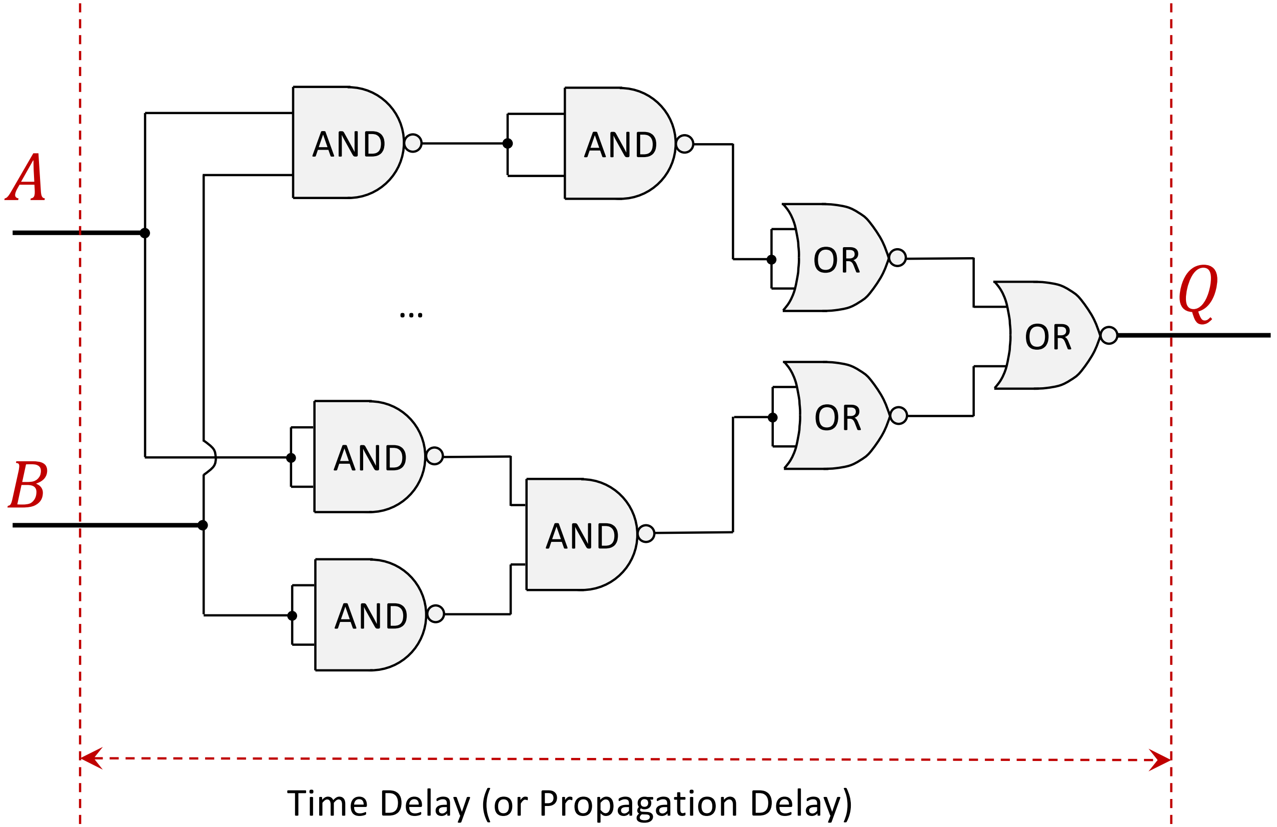
Synchronous

A clock signal governs a circuit operation

CPU registers

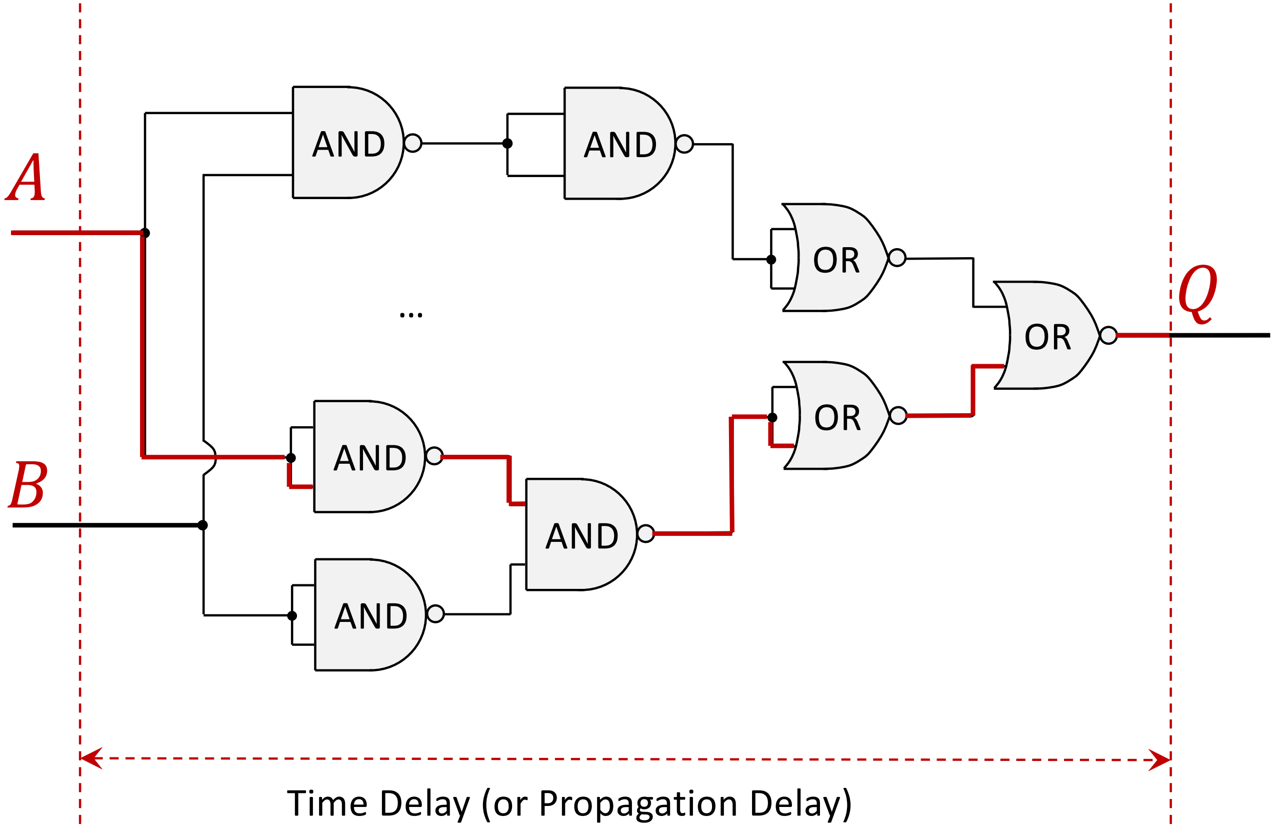
Propagation Delay

There is ALWAYS a time delay between the change of input signals, and the update of an output signal



Critical Path and Propagation Delay

There is ALWAYS a time delay between the change of input signals, and the update of an output signal

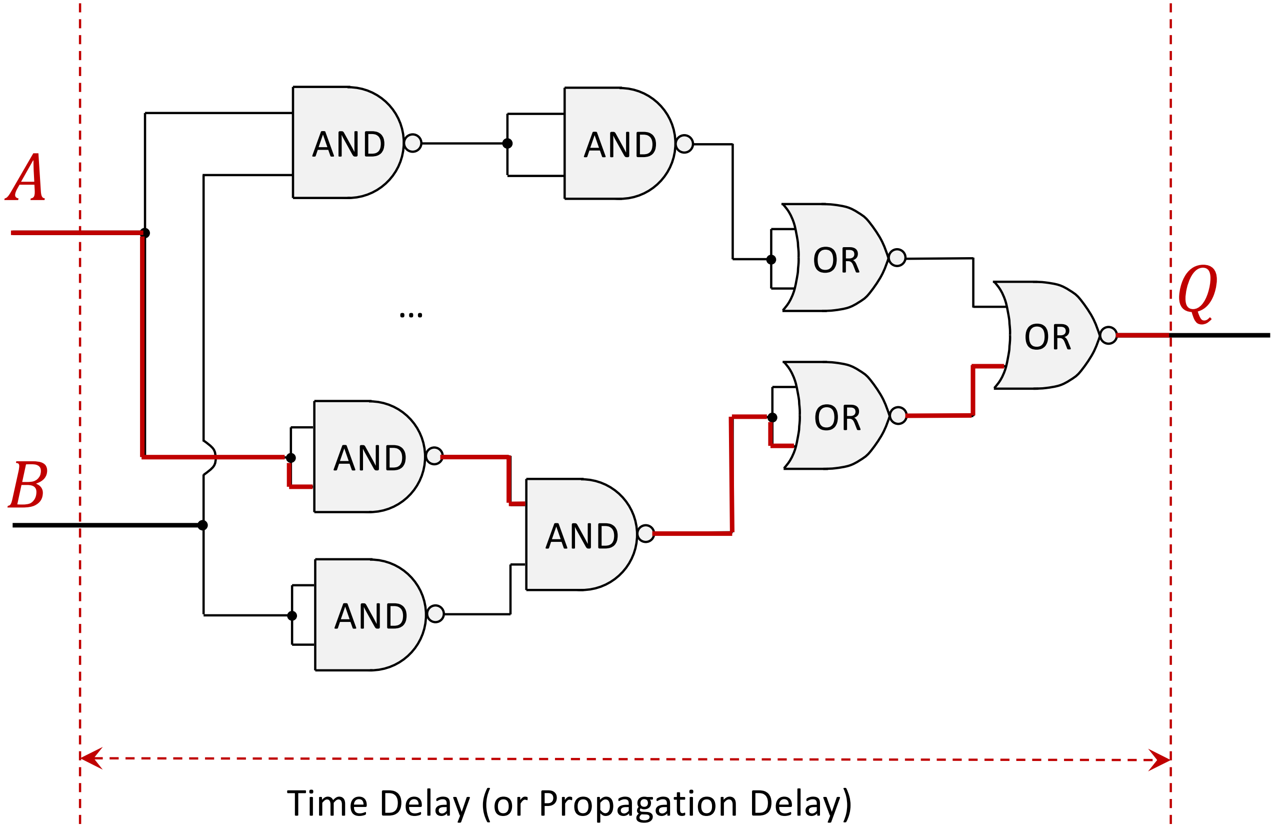


Time Delay (or Propagation Delay)

Critical Path – the one taking the longest time

Critical Path and Propagation Delay

There is ALWAYS a time delay between the change of input signals, and the update of an output signal



Time Delay (or Propagation Delay)

Critical Path – the one taking the longest time