

Computer Architecture
Lecture Week 7 (Part I)

Clock-Driven Circuits

Flip-Flops

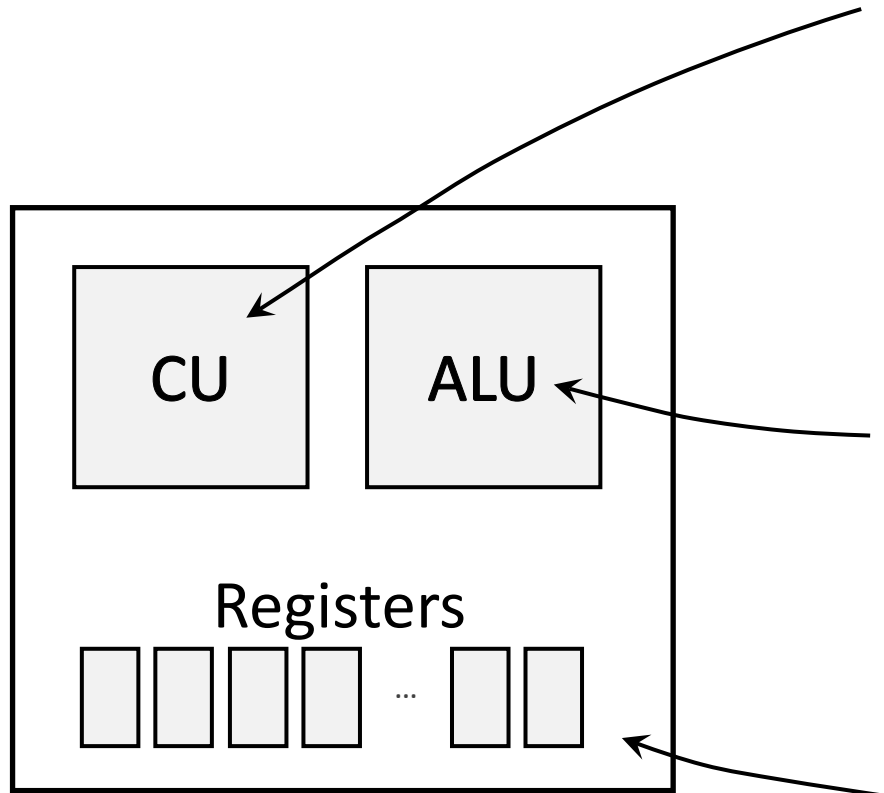
Artem Burmyakov

October 17, 2025



INNOPOLIS
UNIVERSITY

Recap: Key CPU Components



Control Unit (CU):

- 1) To access system memory and cache;
- 2) To synchronise instructions execution;
- 3) Has special-purpose registers inside;
- 4) Other

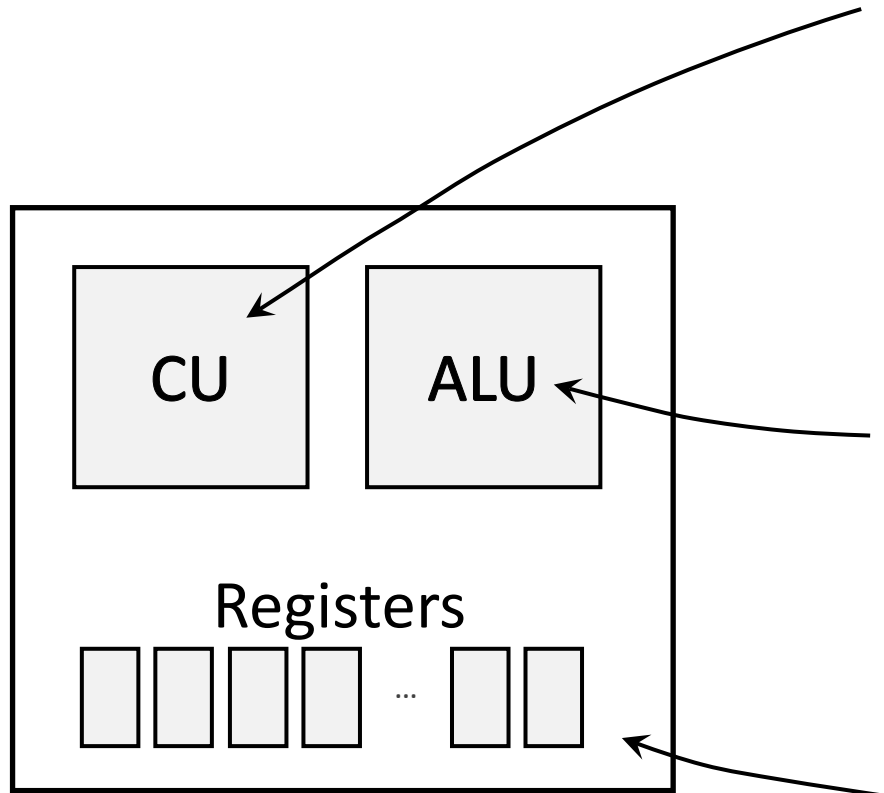
Arithmetic Logic Unit (ALU):

- 1) To implement arithmetic and logic instructions;
- 2) Has no internal registers

Register File:

- 1) To synchronise input arguments provision to ALU;
- 2) To decode instructions;
- 3) To store ALU output

CPU Components: Circuit Types



Control Unit (CU): **Sequential**

- 1) To access system memory and cache;
- 2) To synchronise instructions execution;
- 3) **Has special-purpose registers inside;**
- 4) Other

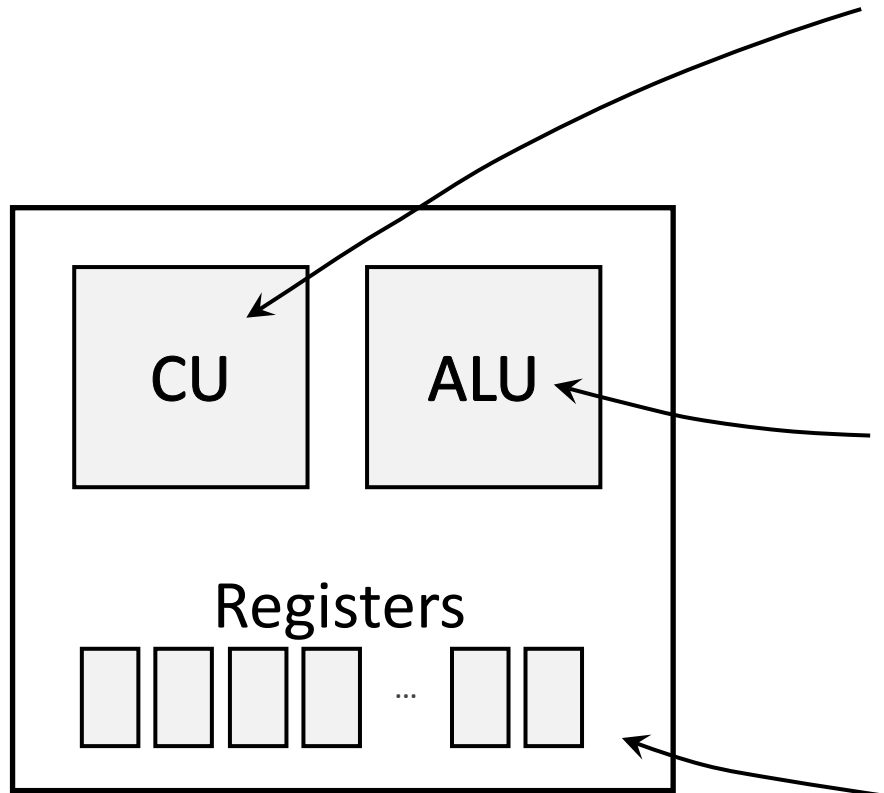
Arithmetic Logic Unit (ALU):

- 1) To implement arithmetic and logic instructions;
- 2) Has no internal registers

Register File:

- 1) To synchronise input arguments provision to ALU;
- 2) To decode instructions;
- 3) To store ALU output

CPU Components: Circuit Types



Control Unit (CU): Sequential

- 1) To access system memory and cache;
- 2) To synchronise instructions execution;
- 3) Has special-purpose registers inside;
- 4) Other

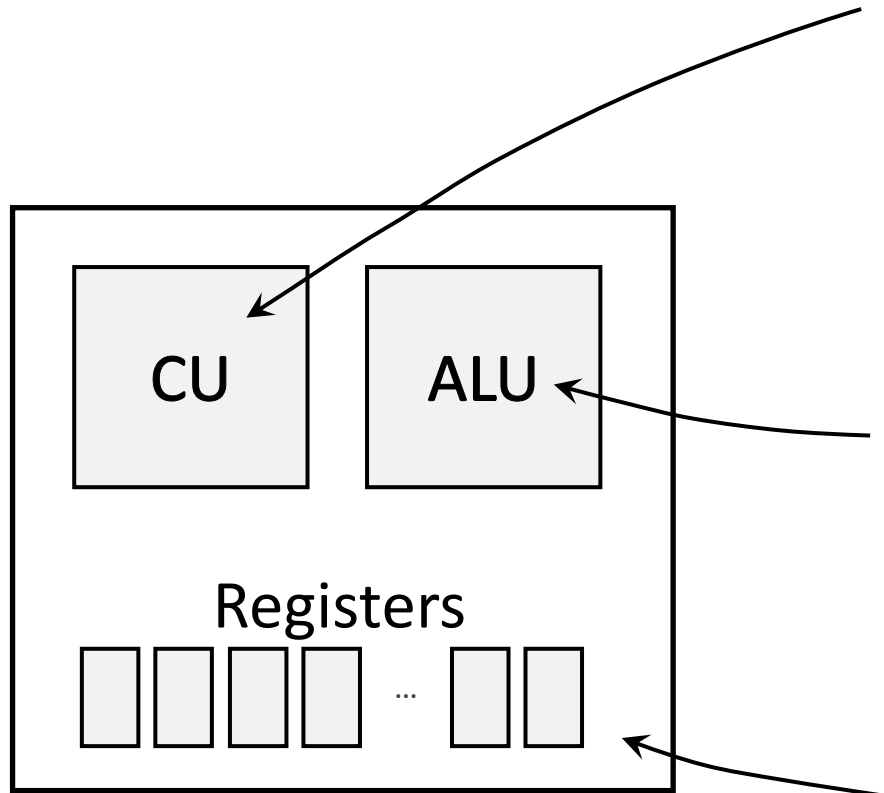
Arithmetic Logic Unit (ALU): Pure Combinational

- 1) To implement arithmetic and logic instructions;
- 2) Has no internal registers

Register File:

- 1) To synchronise input arguments provision to ALU;
- 2) To decode instructions;
- 3) To store ALU output

CPU Components: Circuit Types



Control Unit (CU): Sequential

- 1) To access system memory and cache;
- 2) To synchronise instructions execution;
- 3) Has special-purpose registers inside;
- 4) Other

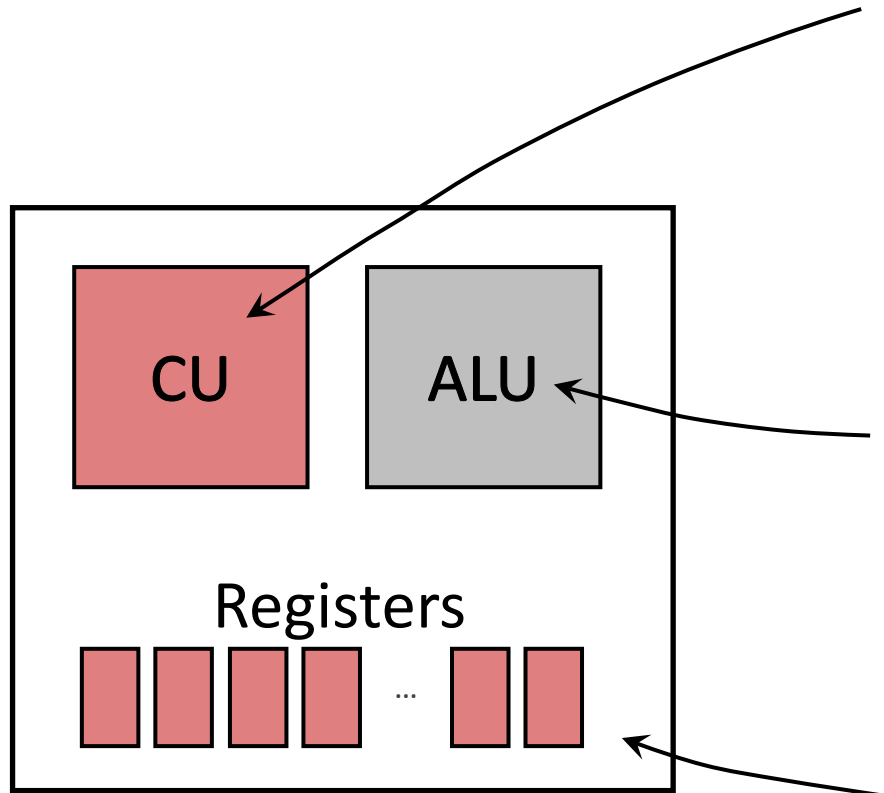
Arithmetic Logic Unit (ALU): Pure Combinational

- 1) To implement arithmetic and logic instructions;
- 2) Has no internal registers

Register File: Sequential

- 1) To synchronise input arguments provision to ALU;
- 2) To decode instructions;
- 3) To store ALU output

CPU Components: Circuit Types



**Rely on “flip-flop” clocked circuits,
which are based on latches**

Control Unit (CU): Sequential

- 1) To access system memory and cache;
- 2) To synchronise instructions execution;
- 3) Has special-purpose registers inside;
- 4) Other

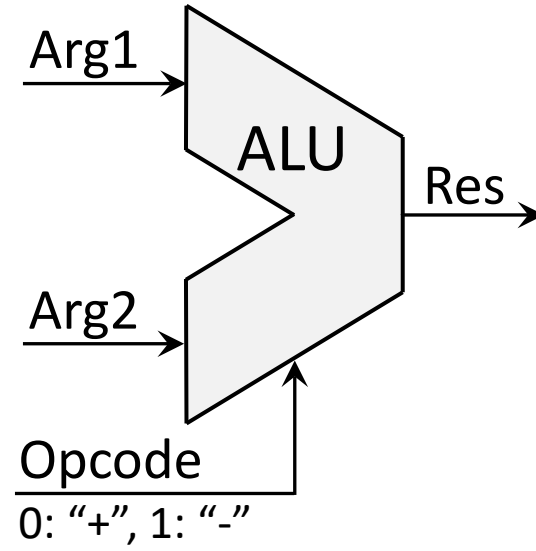
Arithmetic Logic Unit (ALU): Pure Combinational

- 1) To implement arithmetic and logic instructions;
- 2) Has no internal registers

Register File: Sequential

- 1) To synchronise input arguments provision to ALU;
- 2) To decode instructions;
- 3) To store ALU output

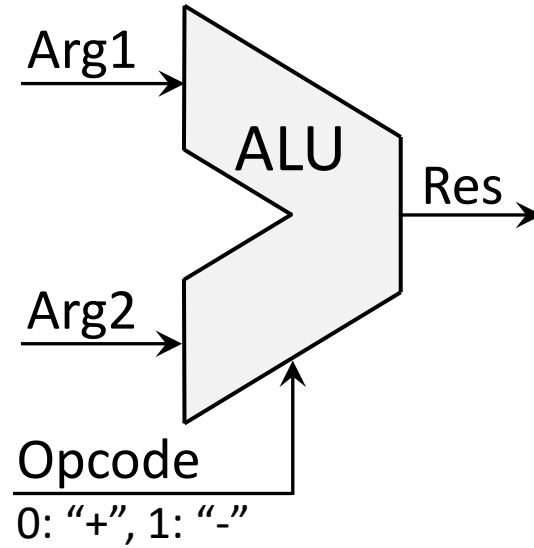
Inputs Synchronization: Motivation



Inputs Synchronization: Motivation

Expected execution:

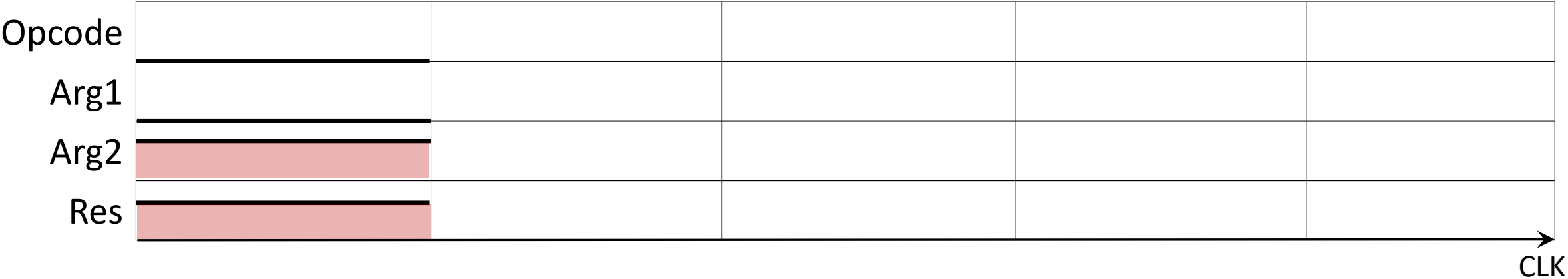
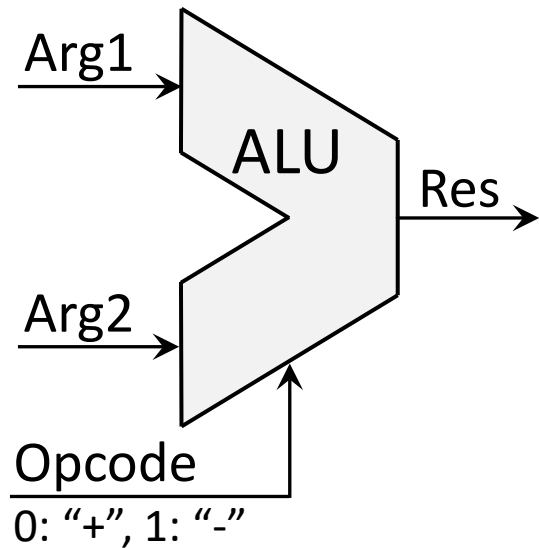
Opcode	Arg1	Arg2	Res (Expected)
0	0	1	1
1	1	1	0
0	0	0	0
0	1	1	10



Inputs Synchronization: Motivation

Expected execution:

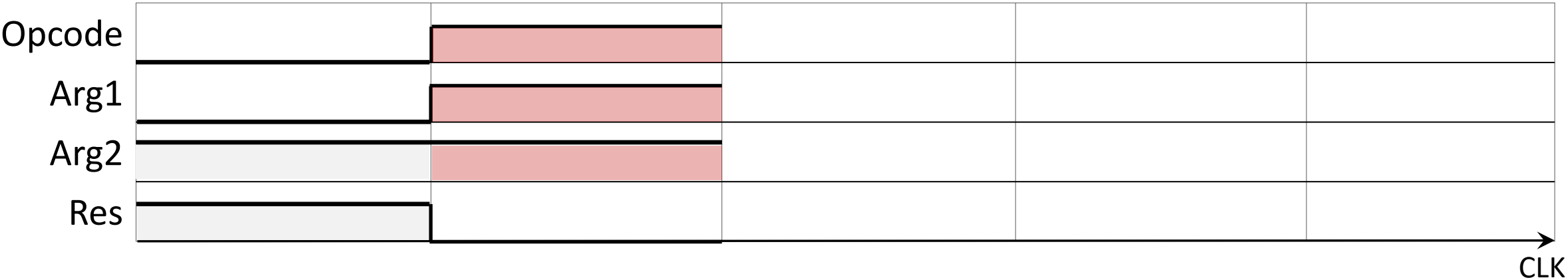
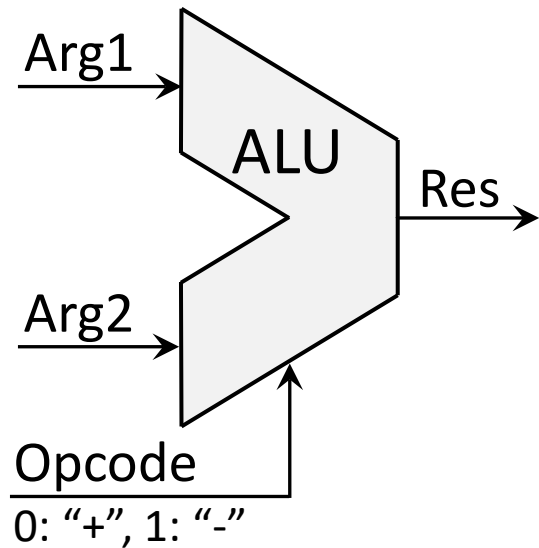
Opcode	Arg1	Arg2	Res (Expected)
0	0	1	1
1	1	1	0
0	0	0	0
0	1	1	10



Inputs Synchronization: Motivation

Expected execution:

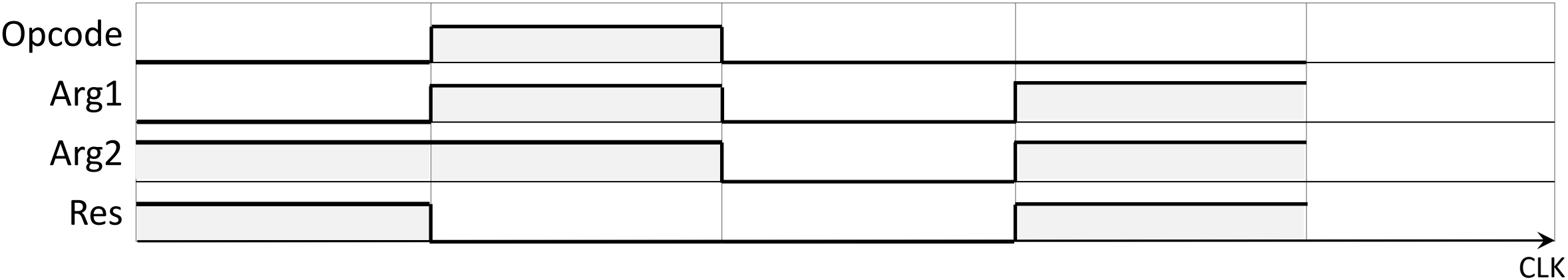
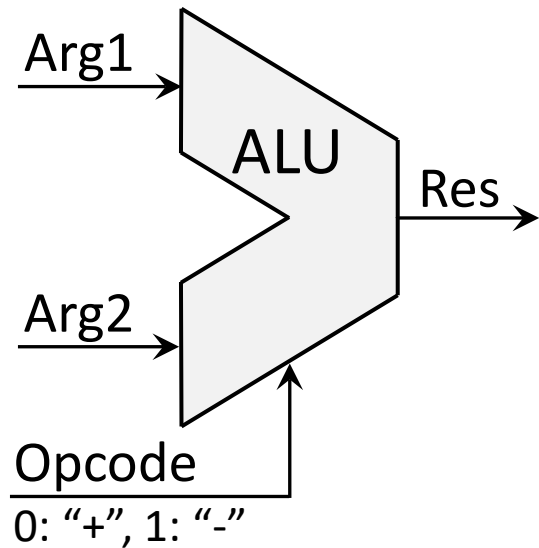
Opcode	Arg1	Arg2	Res (Expected)
0	0	1	1
1	1	1	0
0	0	0	0
0	1	1	10



Inputs Synchronization: Motivation

Expected execution:

Opcode	Arg1	Arg2	Res (Expected)
0	0	1	1
1	1	1	0
0	0	0	0
0	1	1	10



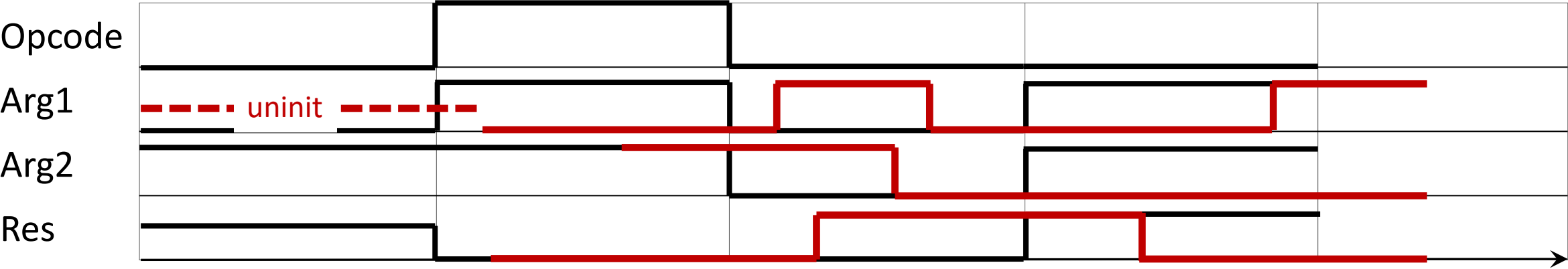
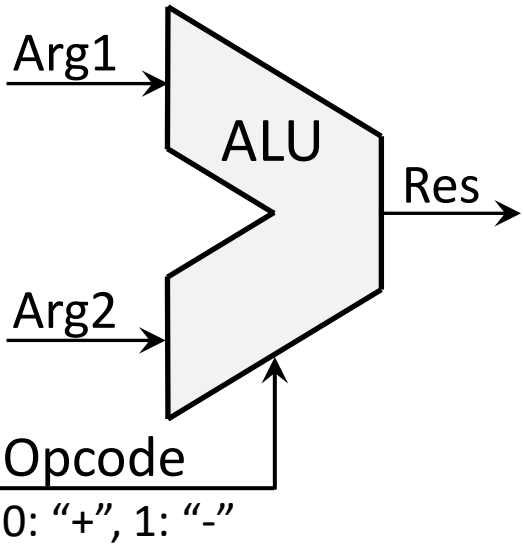
Inputs Synchronization: Motivation

Expected execution:

Opcode	Arg1	Arg2	Res (Expected)
0	0	1	1
1	1	1	0
0	0	0	0
0	1	1	10

Reality (inputs arrive async.):

Opcode	Arg1	Arg2	Res (Reality)
0	uninit	1	uninit
1	0	1	01
0	1	1	10
0	0	1	1



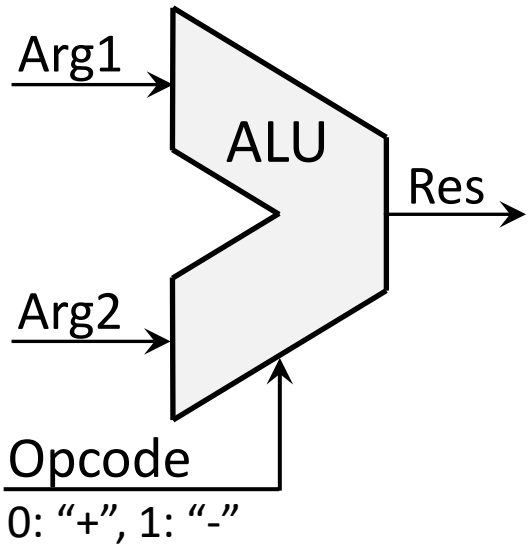
Inputs Synchronization: Motivation

Expected execution:

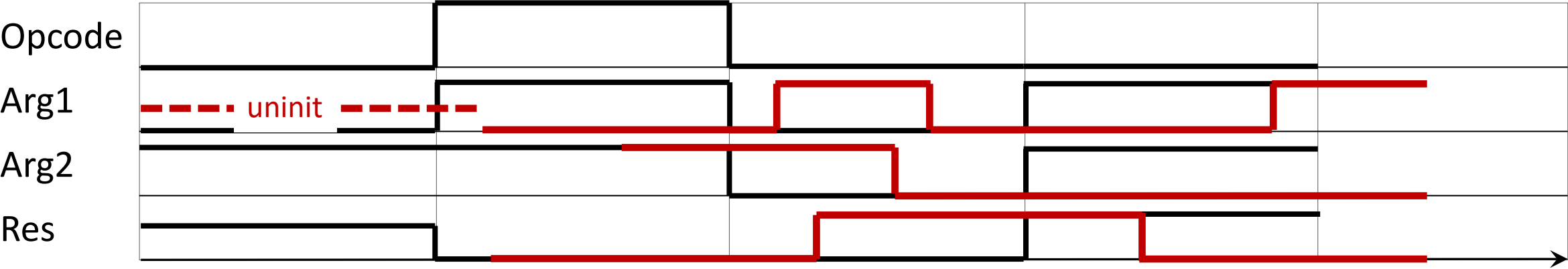
Opcode	Arg1	Arg2	Res (Expected)
0	0	1	1
1	1	1	0
0	0	0	0
0	1	1	10

Reality (inputs arrive async.):

Opcode	Arg1	Arg2	Res (Reality)
0	uninit	1	uninit
1	0	1	01
0	1	1	10
0	0	1	1

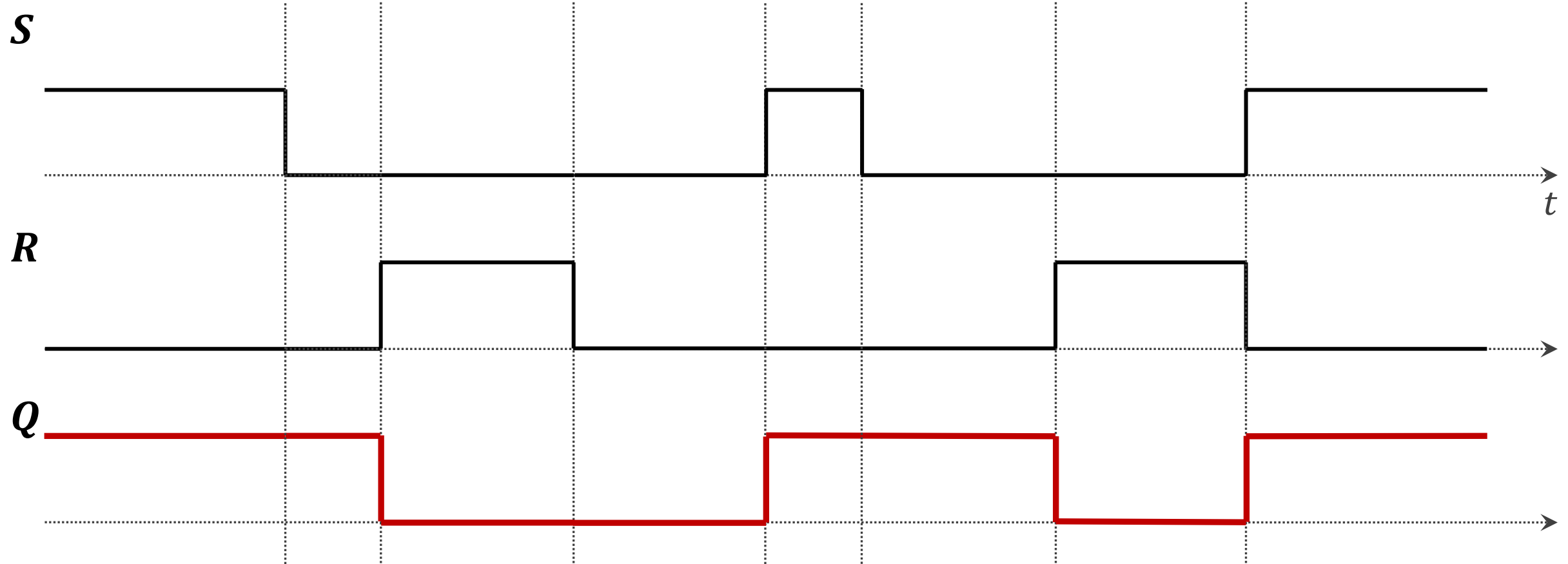


Inputs synchronization is needed!



S/R Latch Behavior Analysis

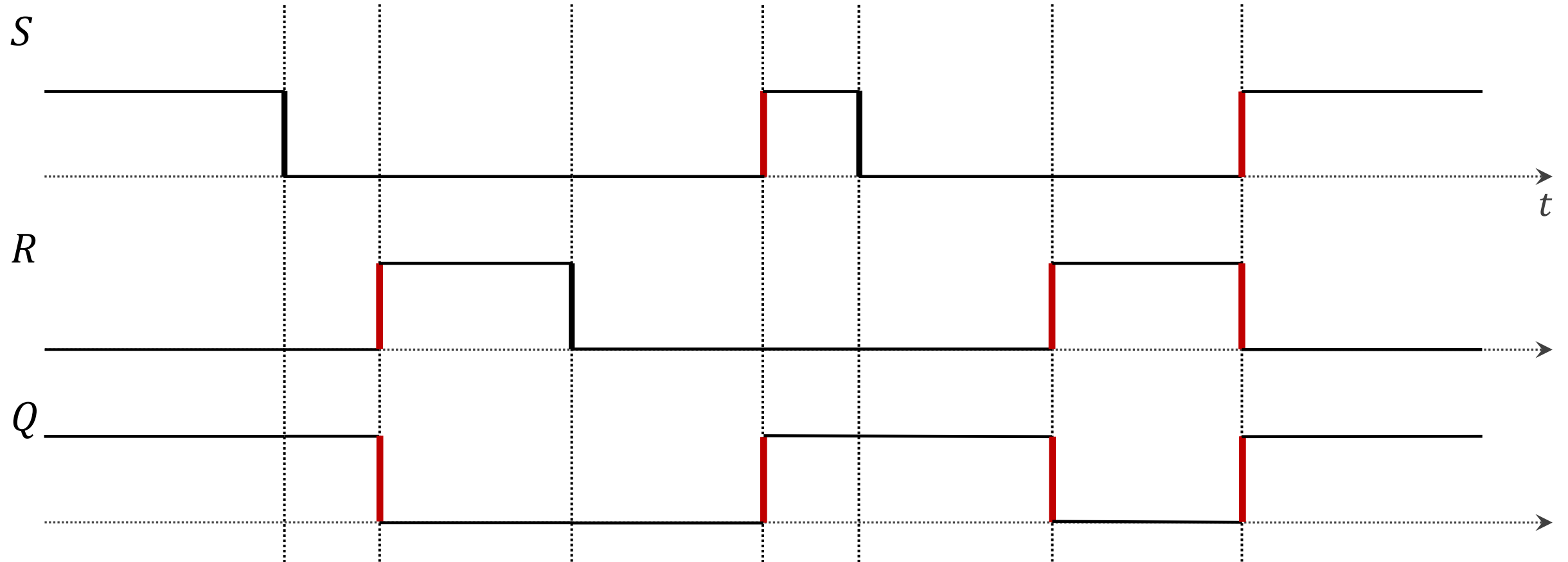
S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal



S/R Latch Inputs Synchronization

No delay between S, R, and Q updates

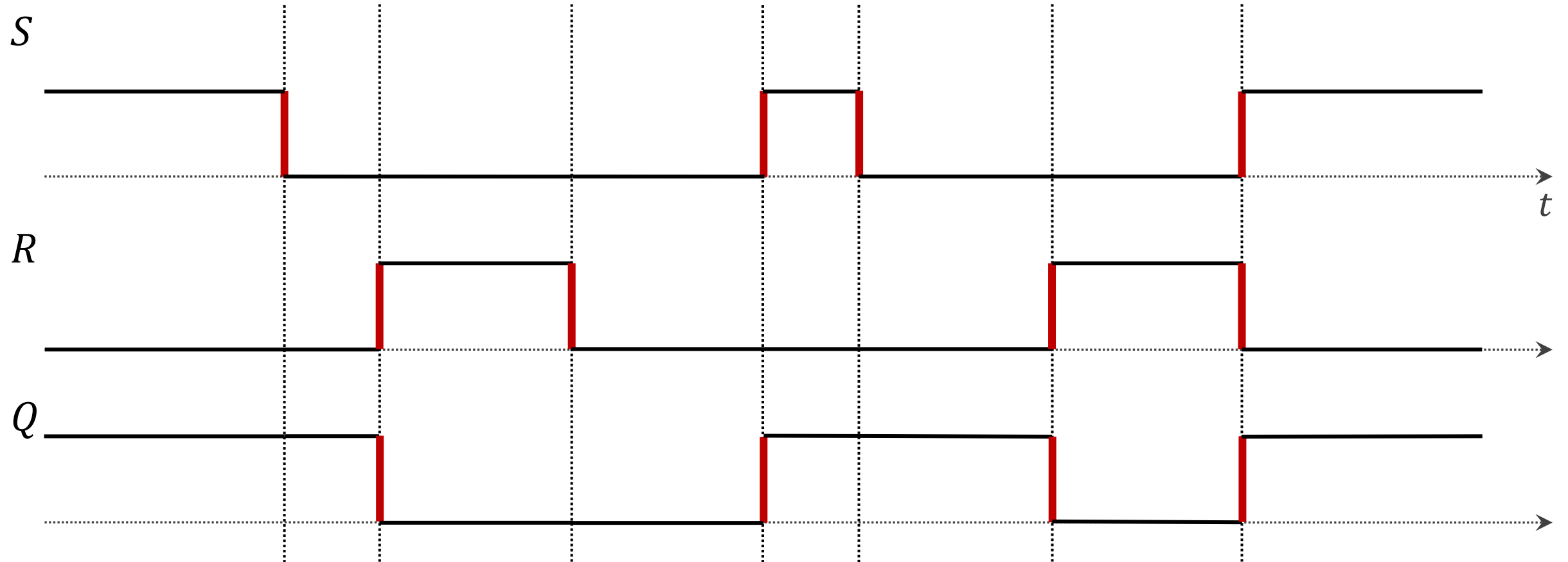
S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal



S/R Latch Inputs Synchronization

- S and R - outputs of other (different) circuits**

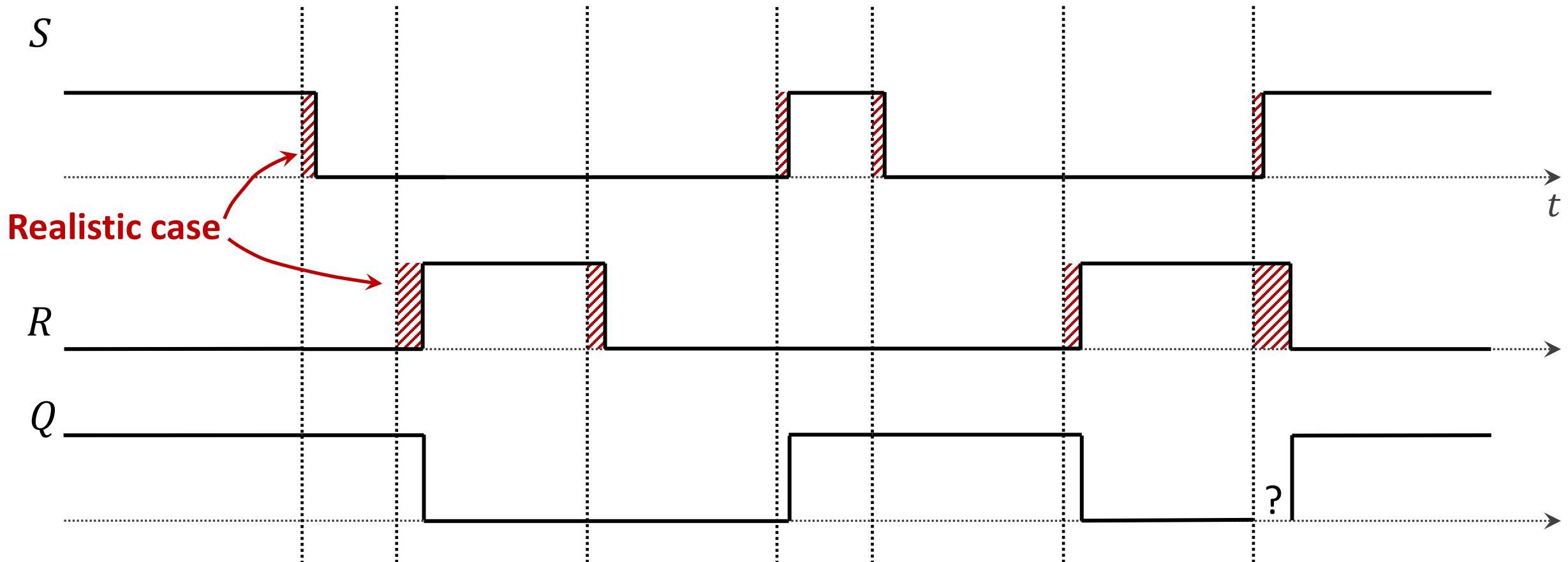
S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal



S/R Latch Inputs Synchronization

- S and R - outputs of other (different) circuits;
- **S and R update times are not synchronised, due to propagation delays**

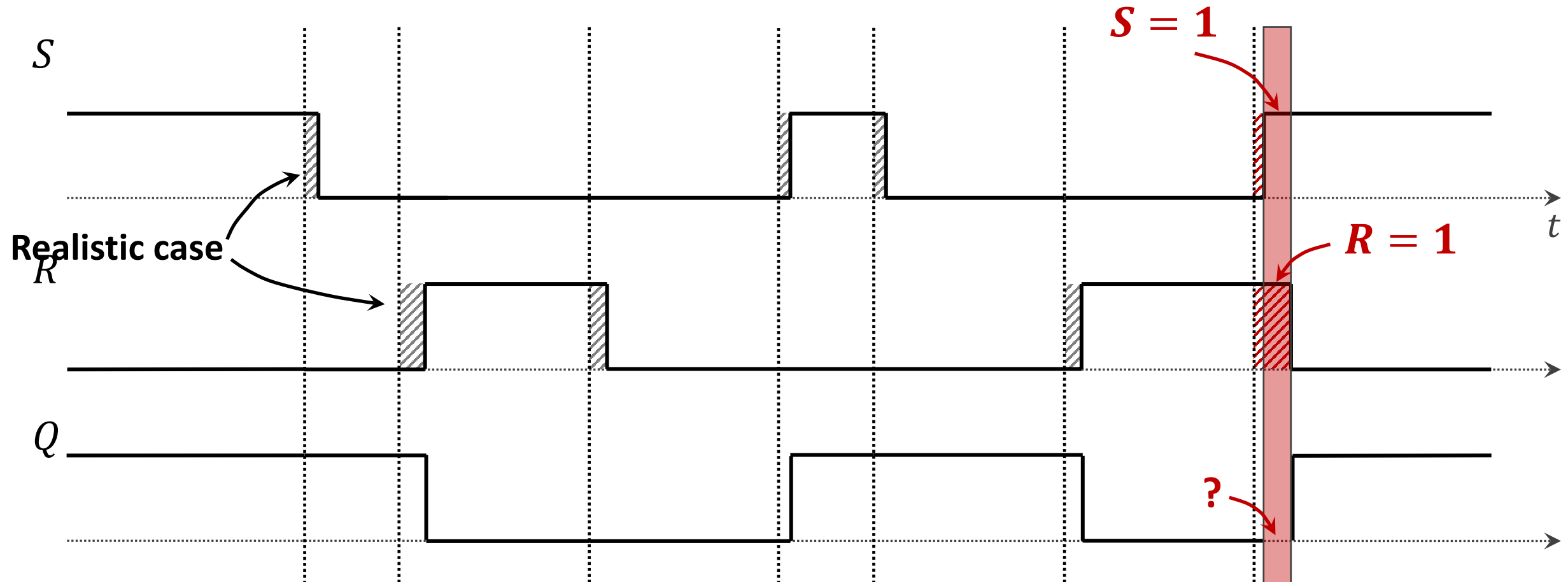
S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal



S/R Latch Inputs Synchronization

- S and R - outputs of other (different) circuits;
- **S and R update times are not synchronised, due to propagation delays**

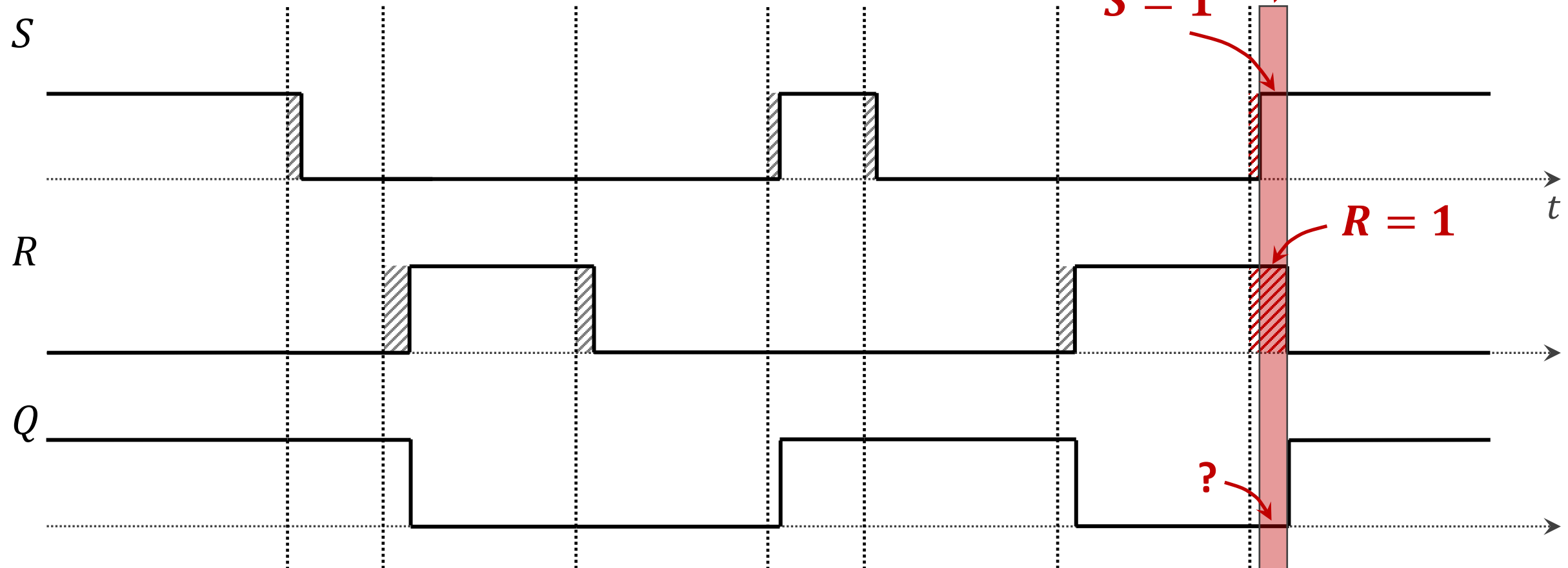
S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal



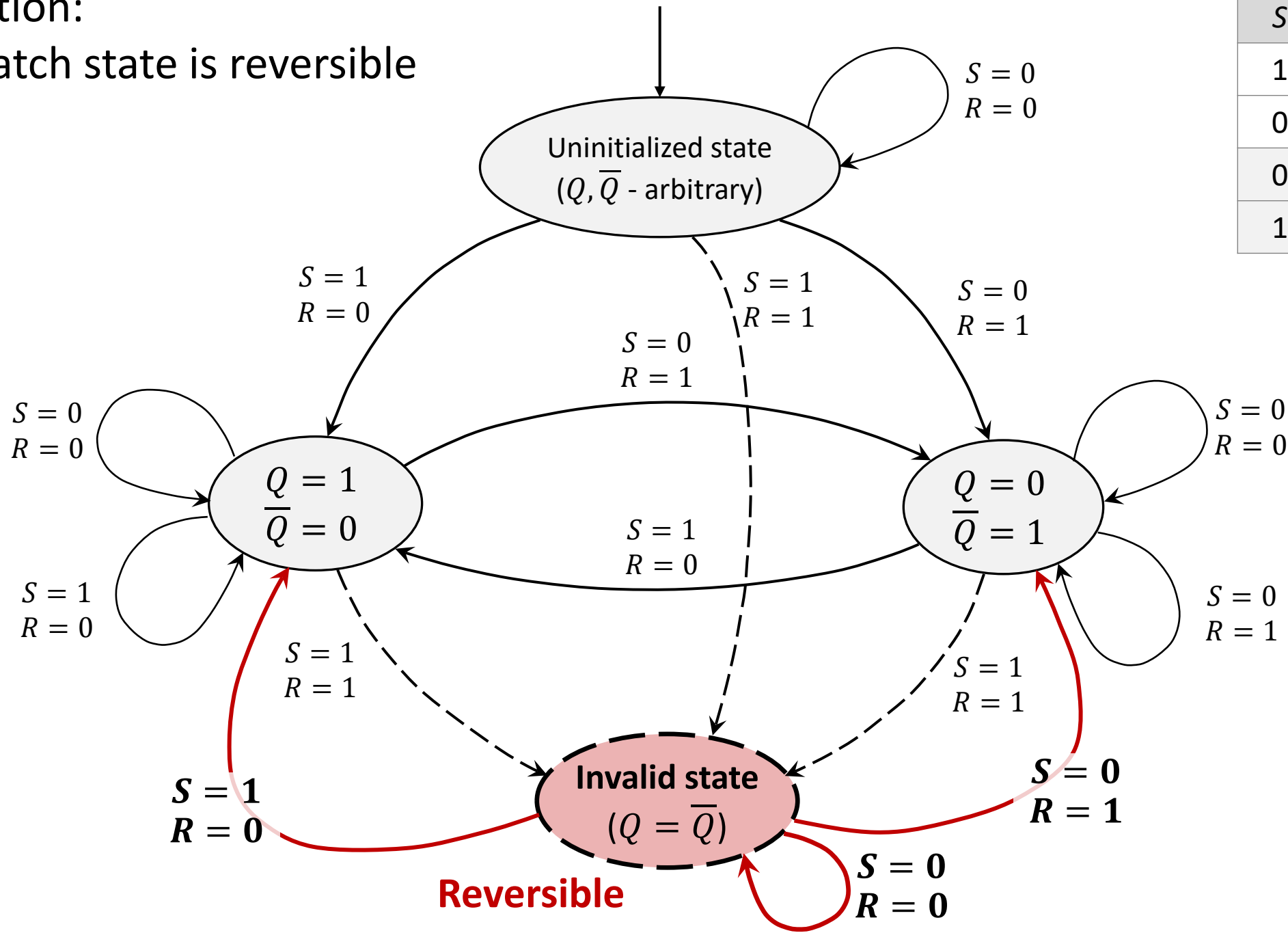
S/R Latch Inputs Synchronization

- S and R - outputs of other (different) circuits;
- S and R update times are not synchronised;
- **Invalid inputs combination is possible**

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

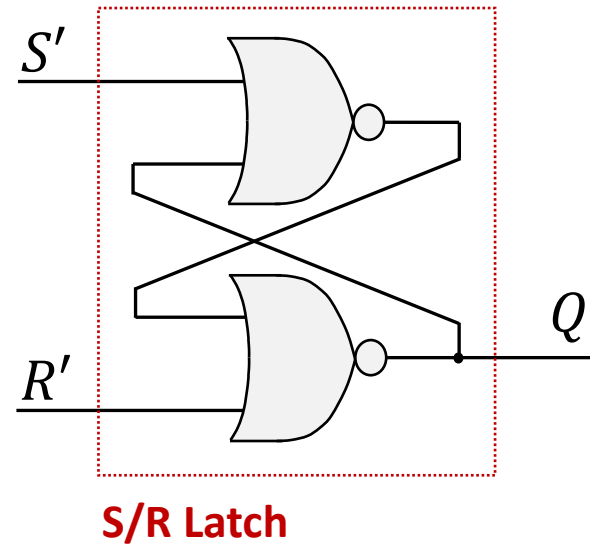


Assumption:
Invalid latch state is reversible



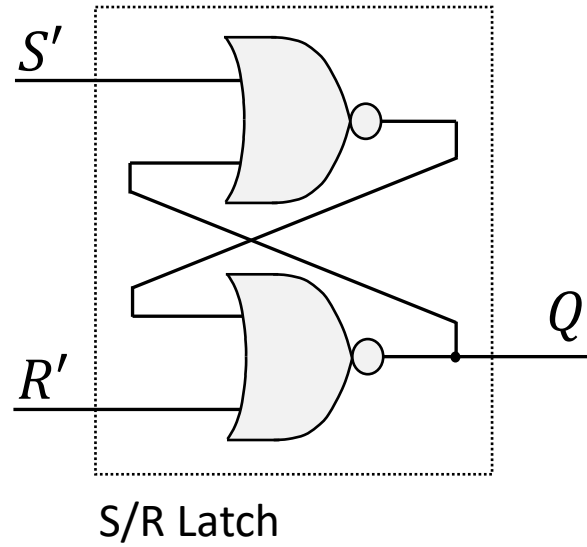
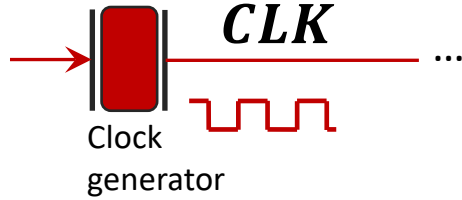
S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal

Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



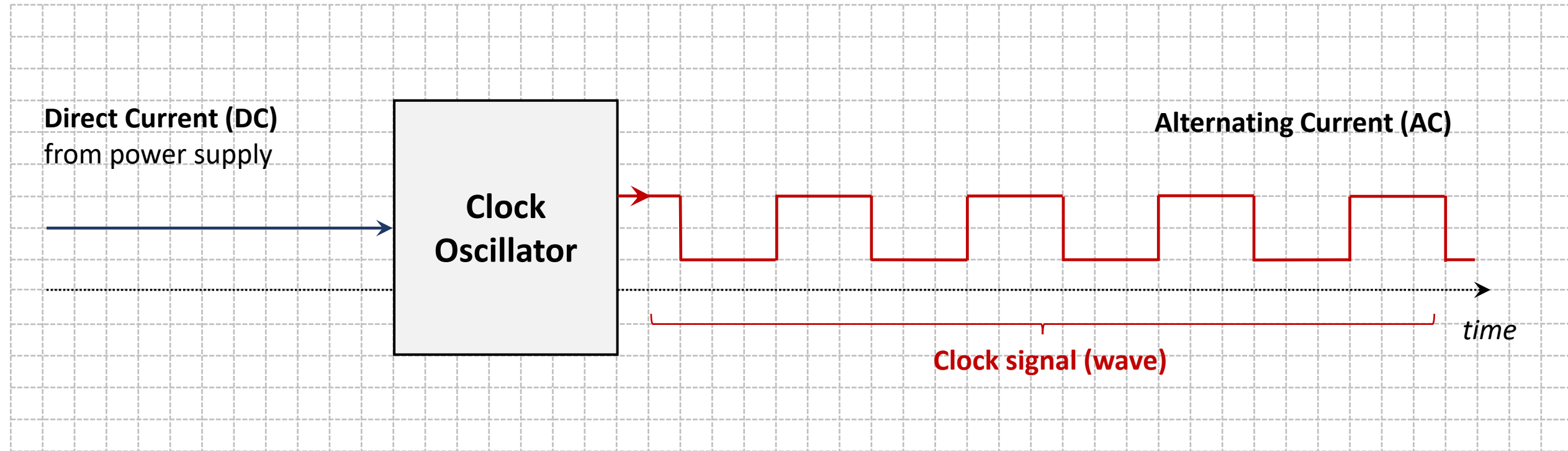
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

Step 1:
The inclusion
of a clock
generator



Clock Generator (Oscillator)

An electronic device, producing a periodic clock signal

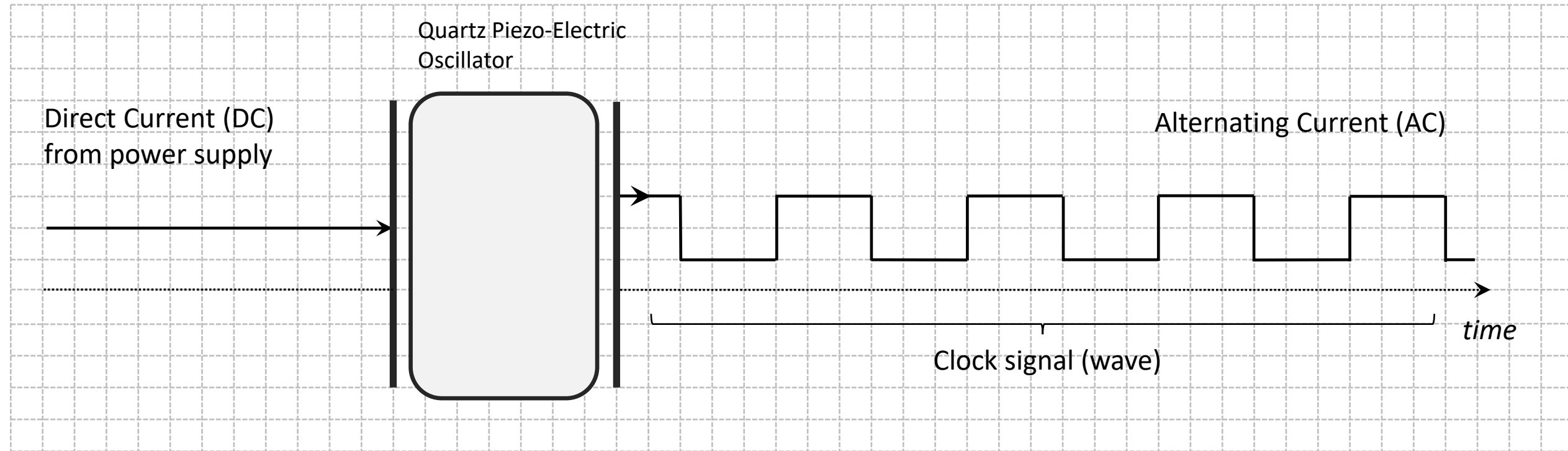


Clock oscillator – a physical device, that converts direct current (DC) from a power supply into an alternating periodic signal

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

Clock Generator (Oscillator)

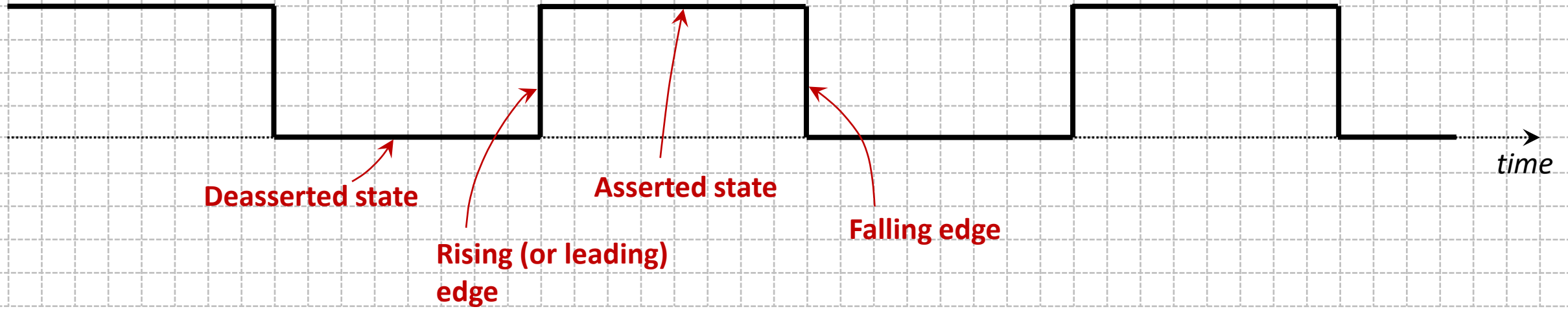
An electronic device, producing a periodic clock signal



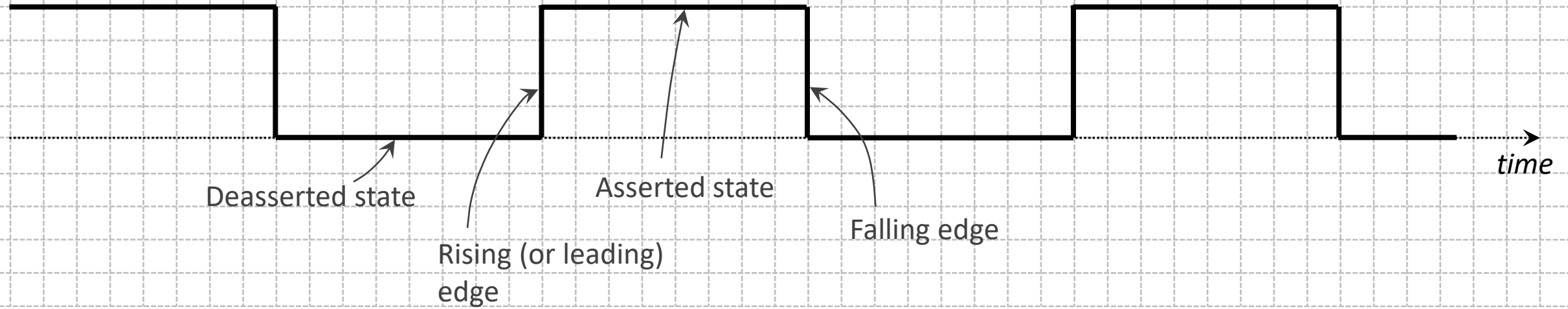
Clock oscillator – a physical device, that converts direct current (DC) from a power supply into an alternating periodic signal

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

Clock Signal Terminology



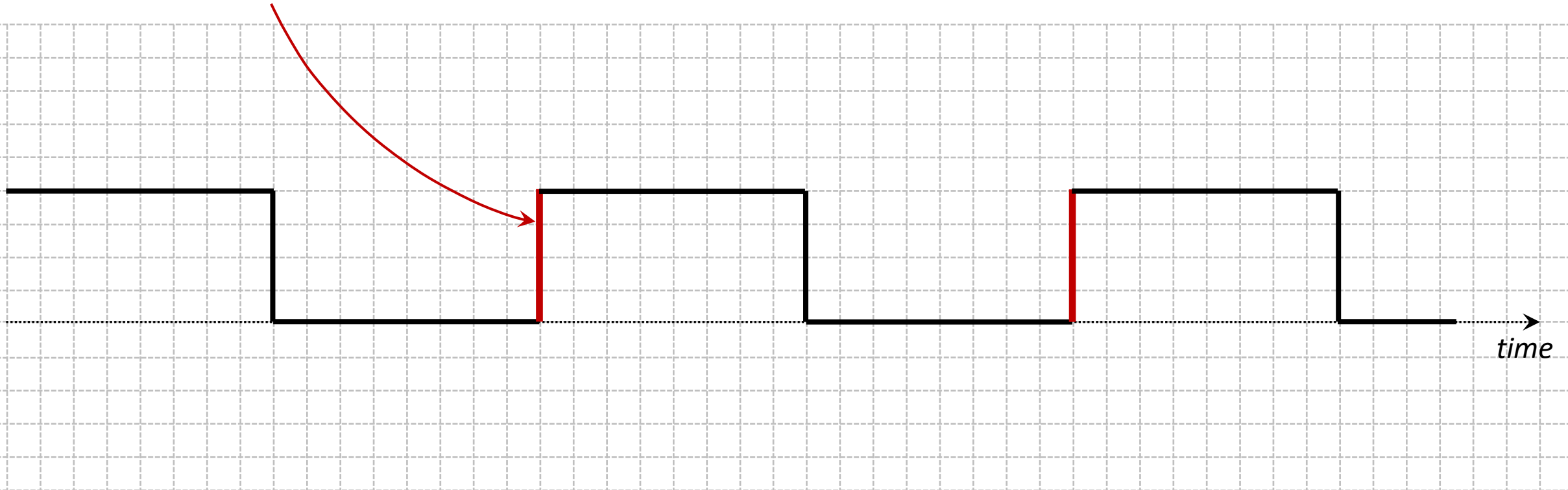
Edge-Triggered and Level-Triggered Timing Methodologies



Edge-Triggered and Level-Triggered Timing Methodologies

Edge-Triggered Circuit:

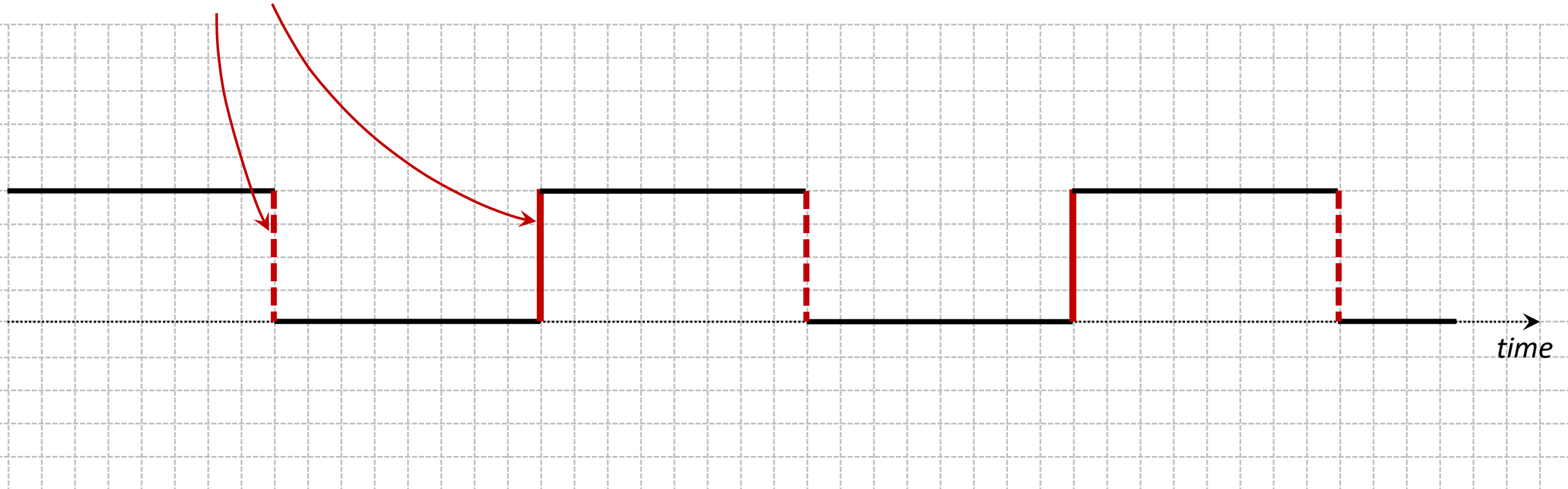
a circuit is activated either on a raising or a falling edge



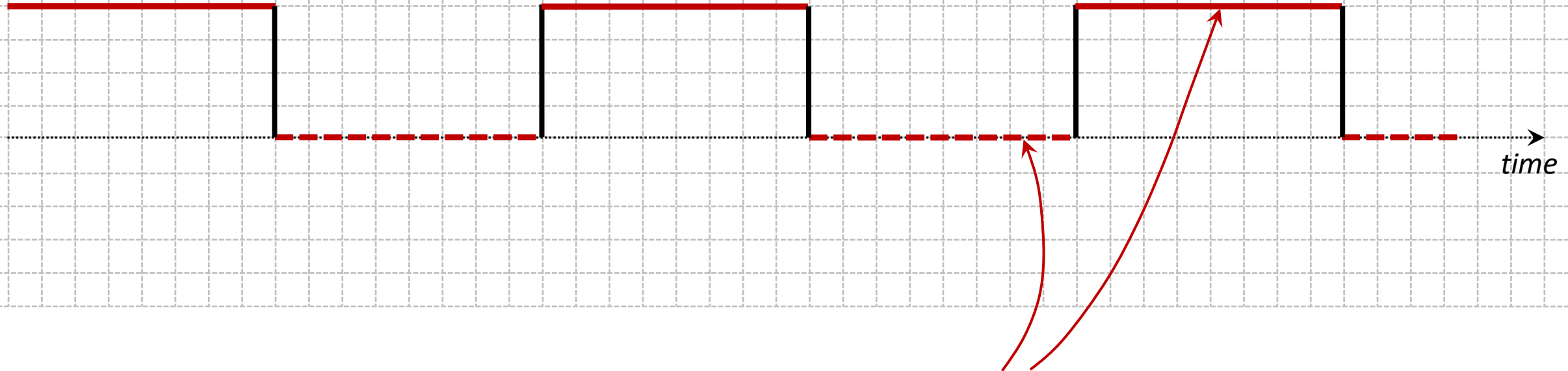
Edge-Triggered and Level-Triggered Timing Methodologies

Edge-Triggered Circuit:

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



Edge-Triggered and Level-Triggered Timing Methodologies



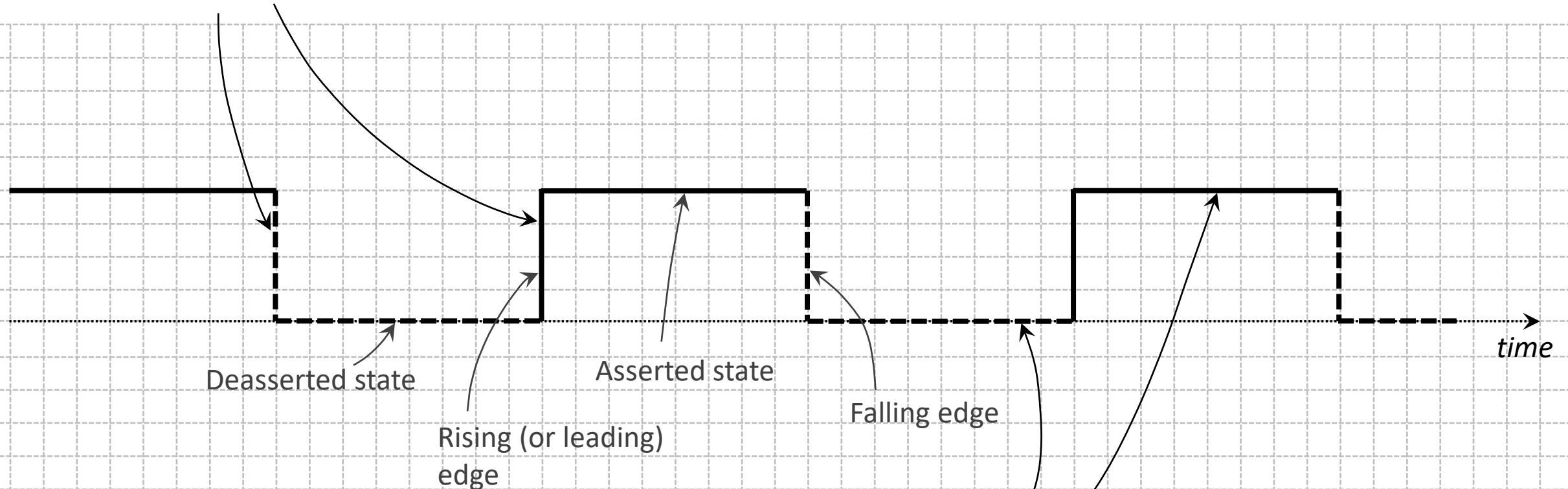
Level-Triggered Circuit:

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

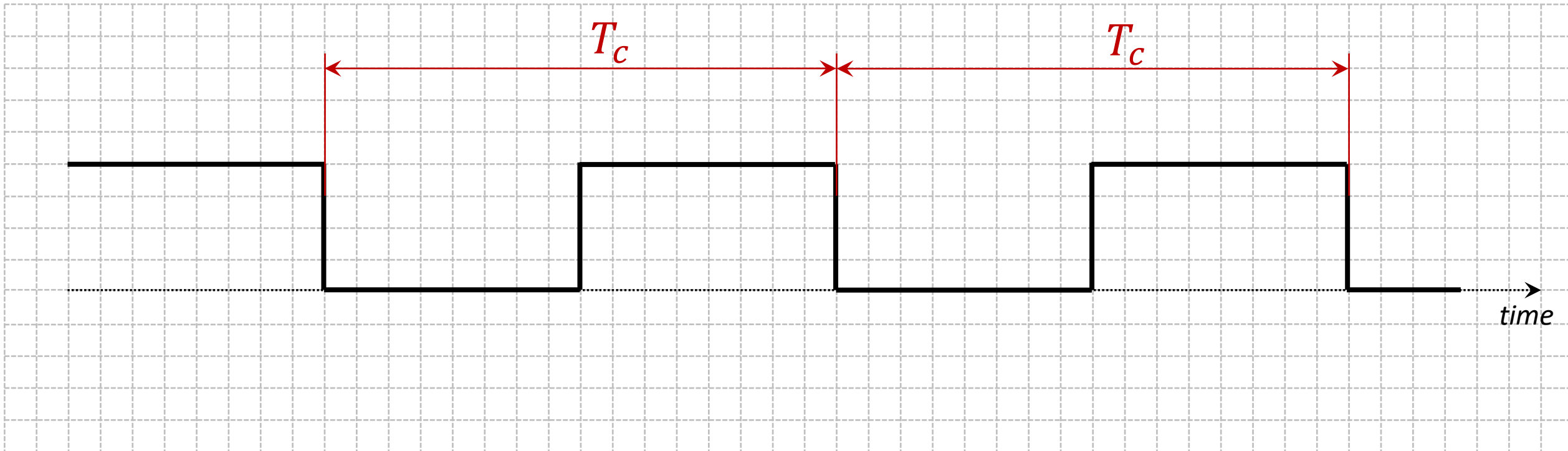


Level-Triggered:

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

Characteristics of a Rectangular Clock Signal

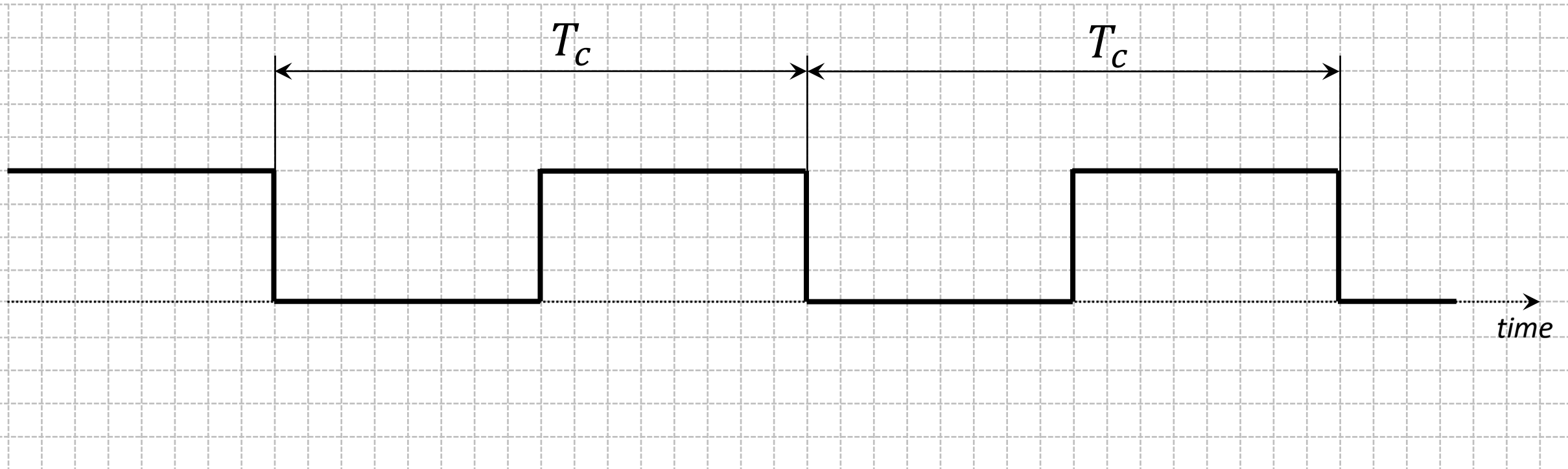
T_c - clock cycle time, or period, or cycle length (measured in ~ns)



Characteristics of a Rectangular Clock Signal

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

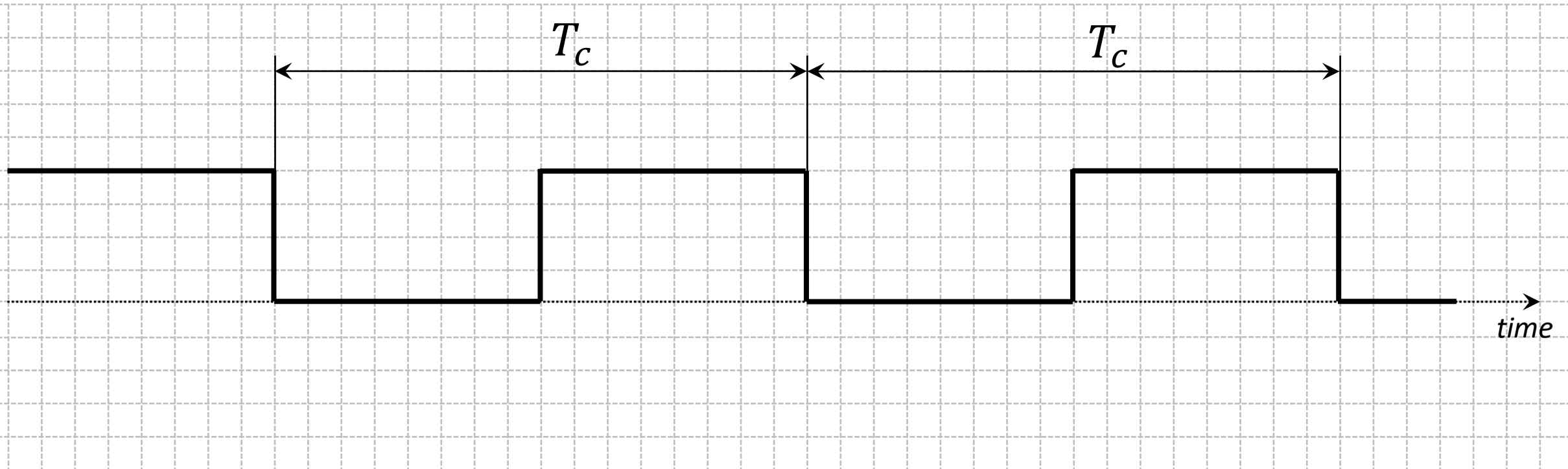
$F_c = 1/T_c$ - clock frequency (measured in ~GHz)



Characteristics of a Rectangular Clock Signal

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

$F_c = 1/T_c$ - clock frequency (measured in ~GHz)

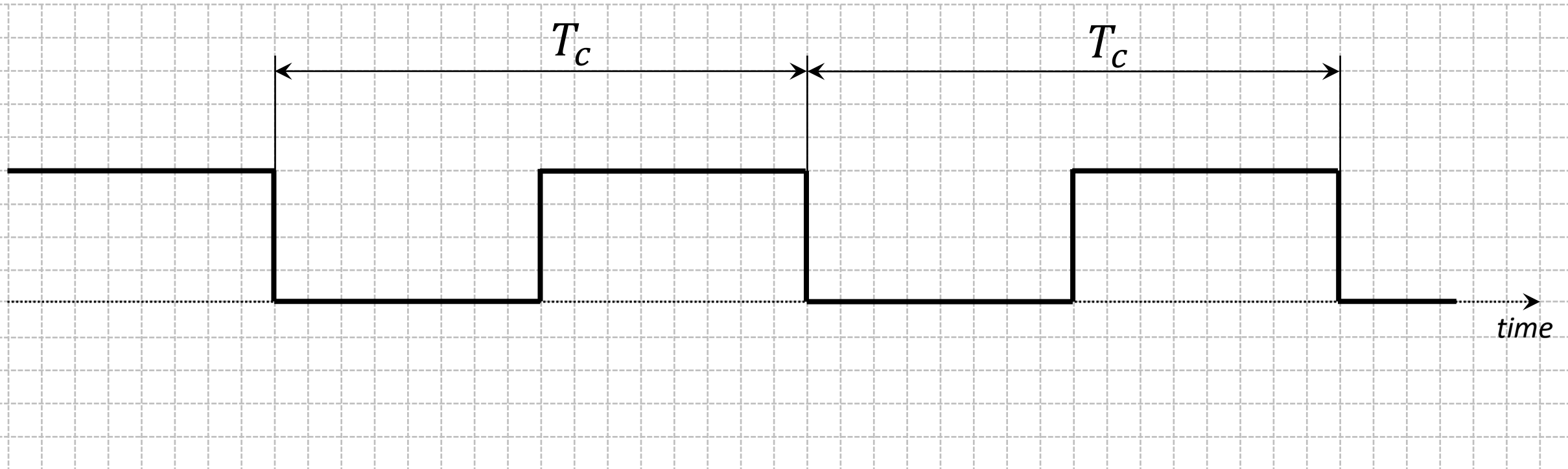


Clock frequency sets the tempo for the processor to execute instructions

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

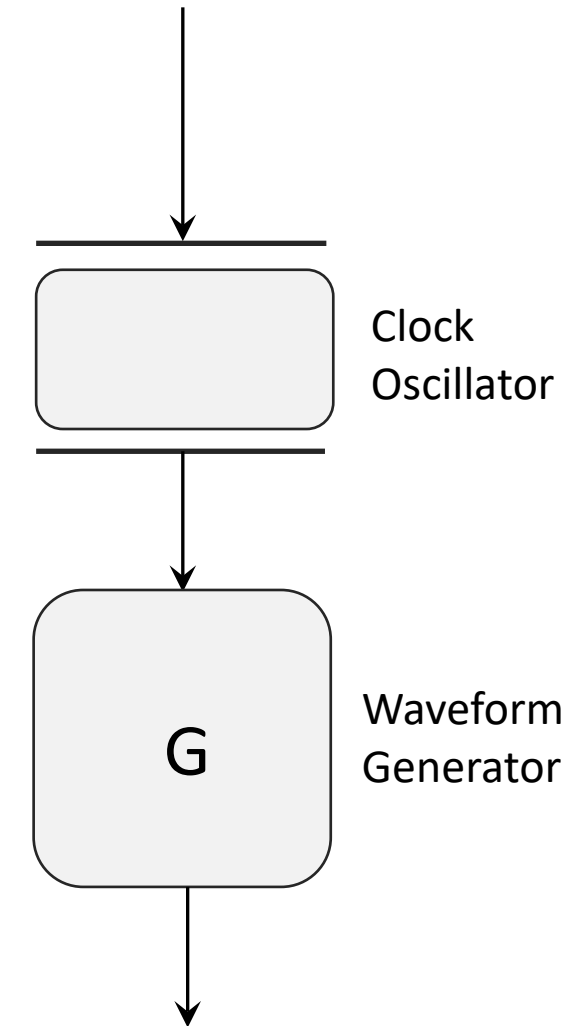
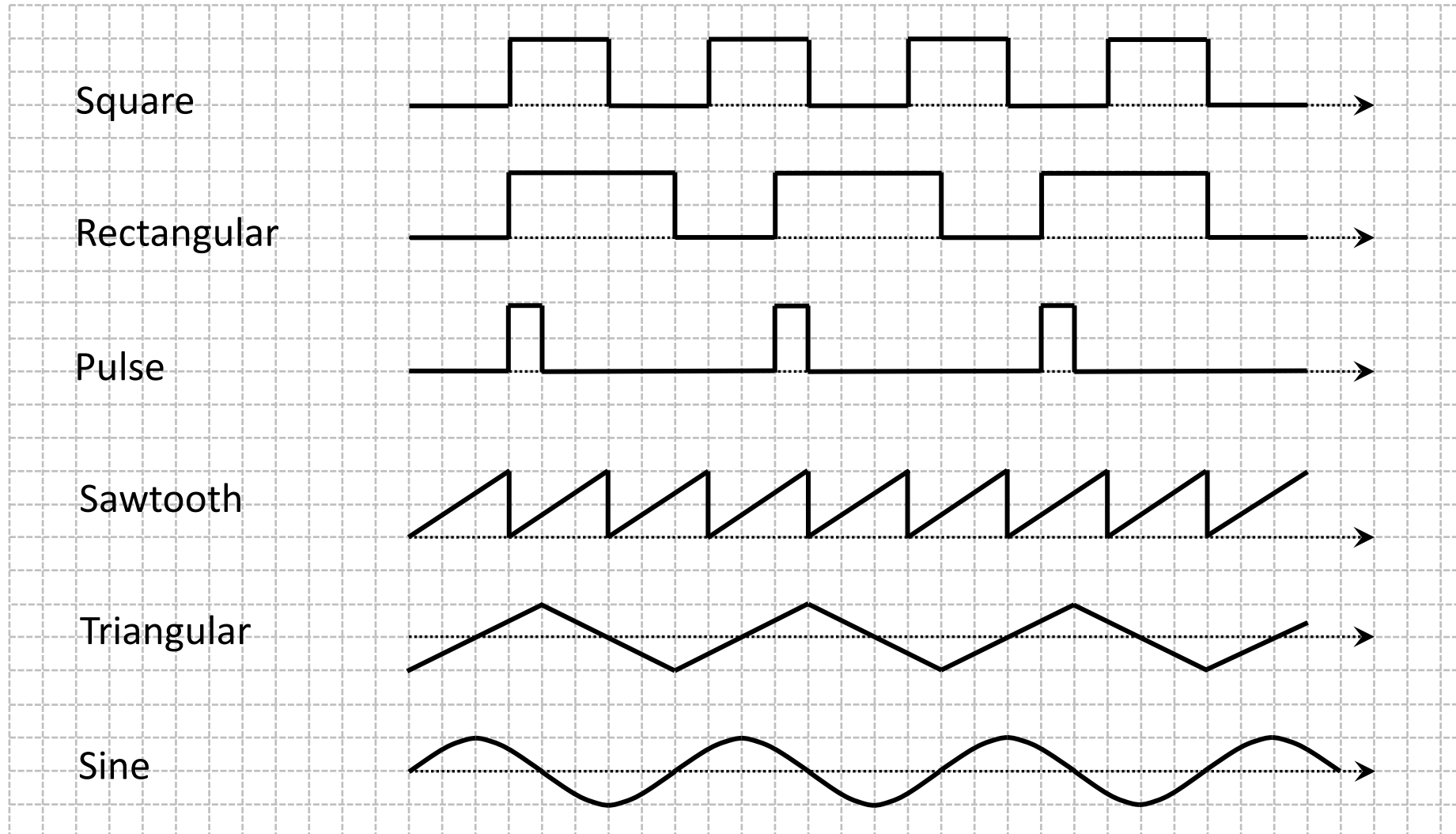
Common characteristics for modern general-purpose CPUs:

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

F_c - a few gigahertz, 2-3 GHz

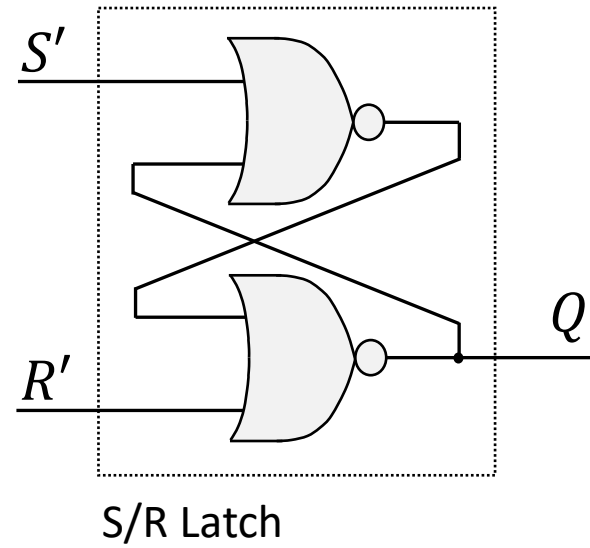
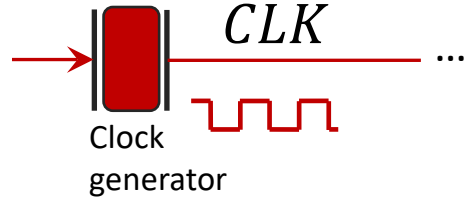
Clock Signal Variations

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

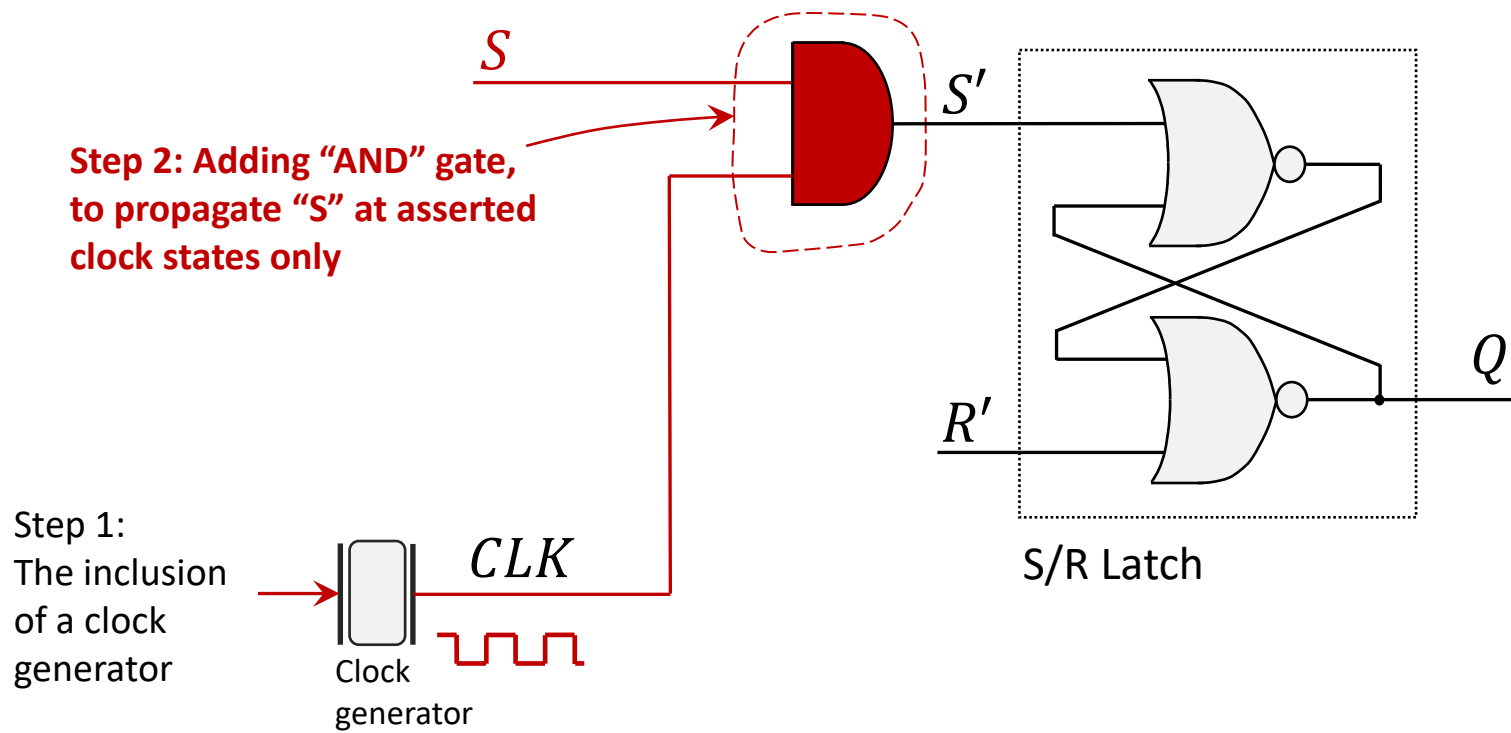


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

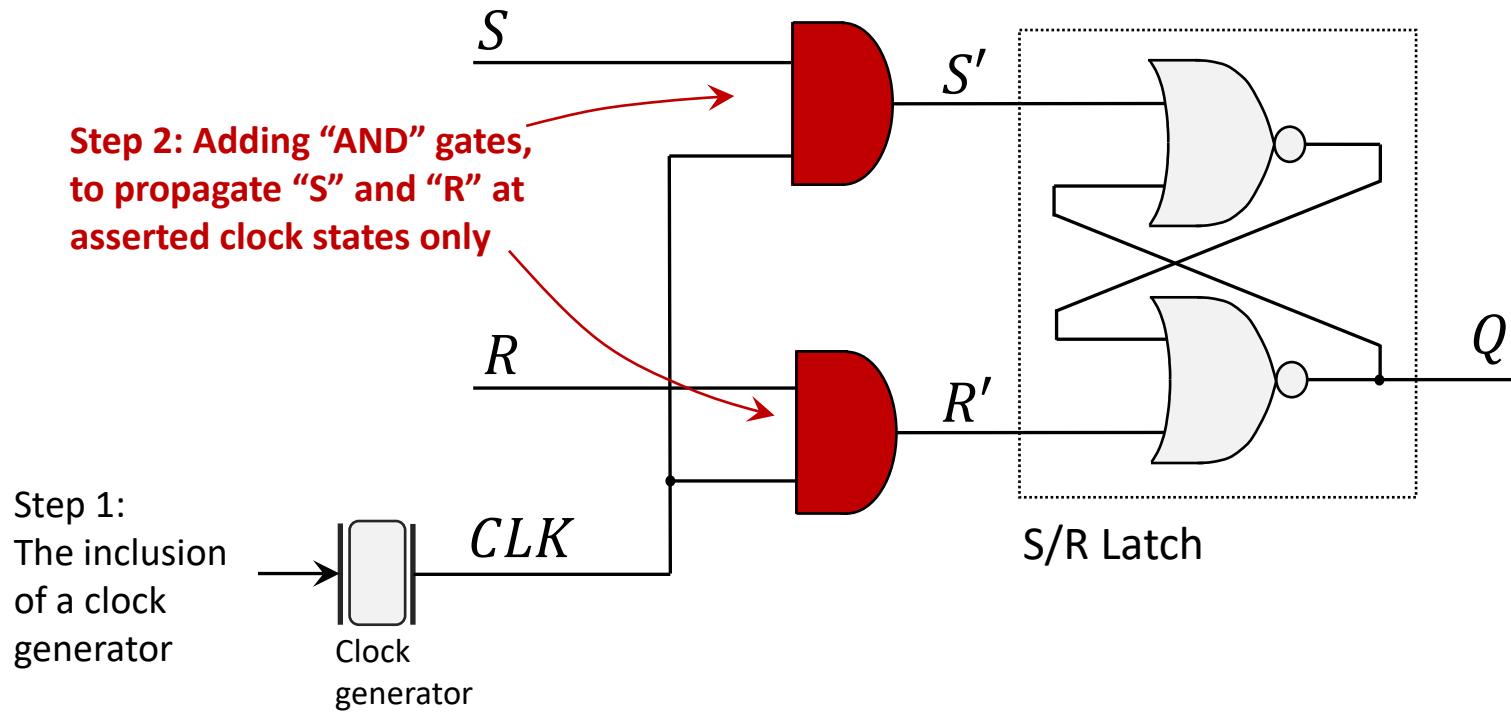
Step 1:
The inclusion
of a clock
generator



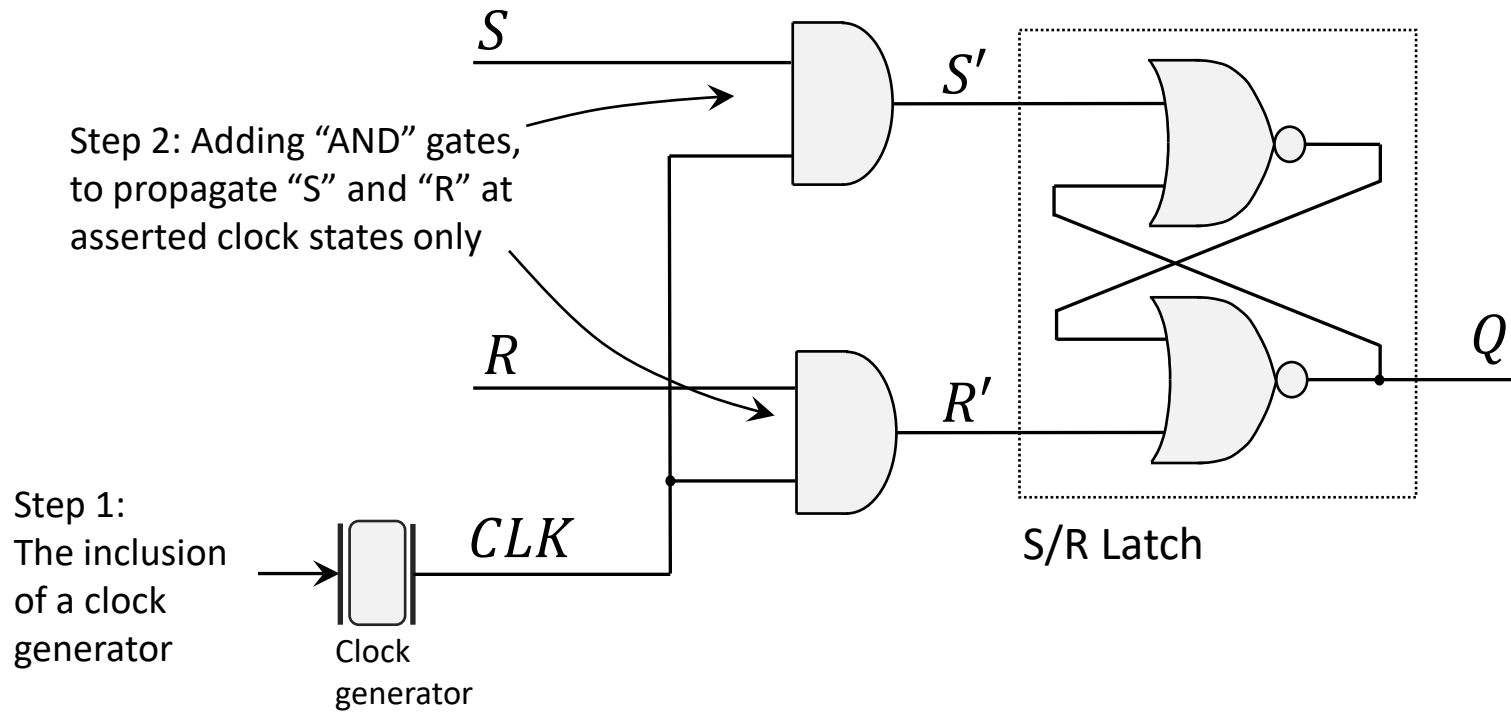
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



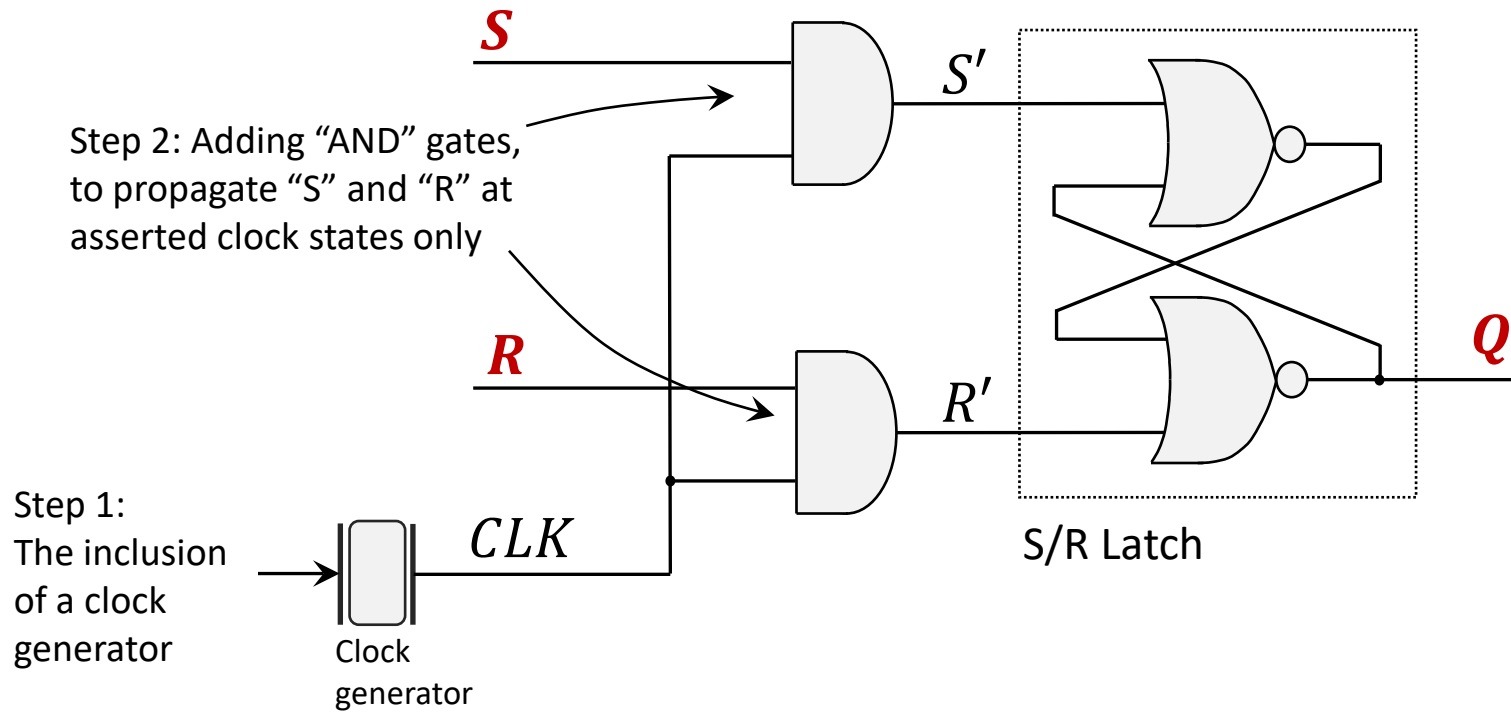
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



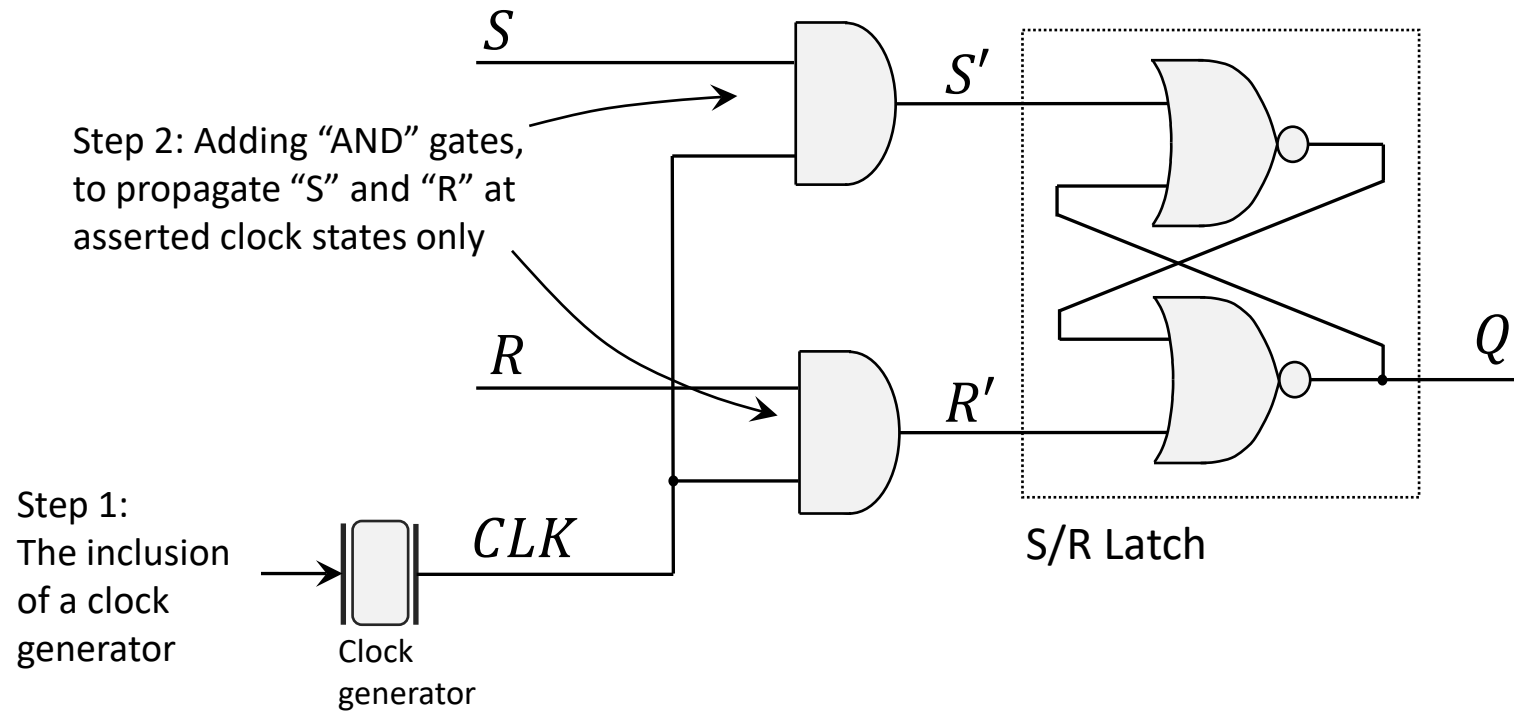
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



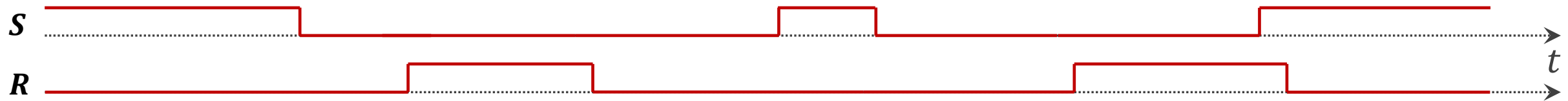
Observation:

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

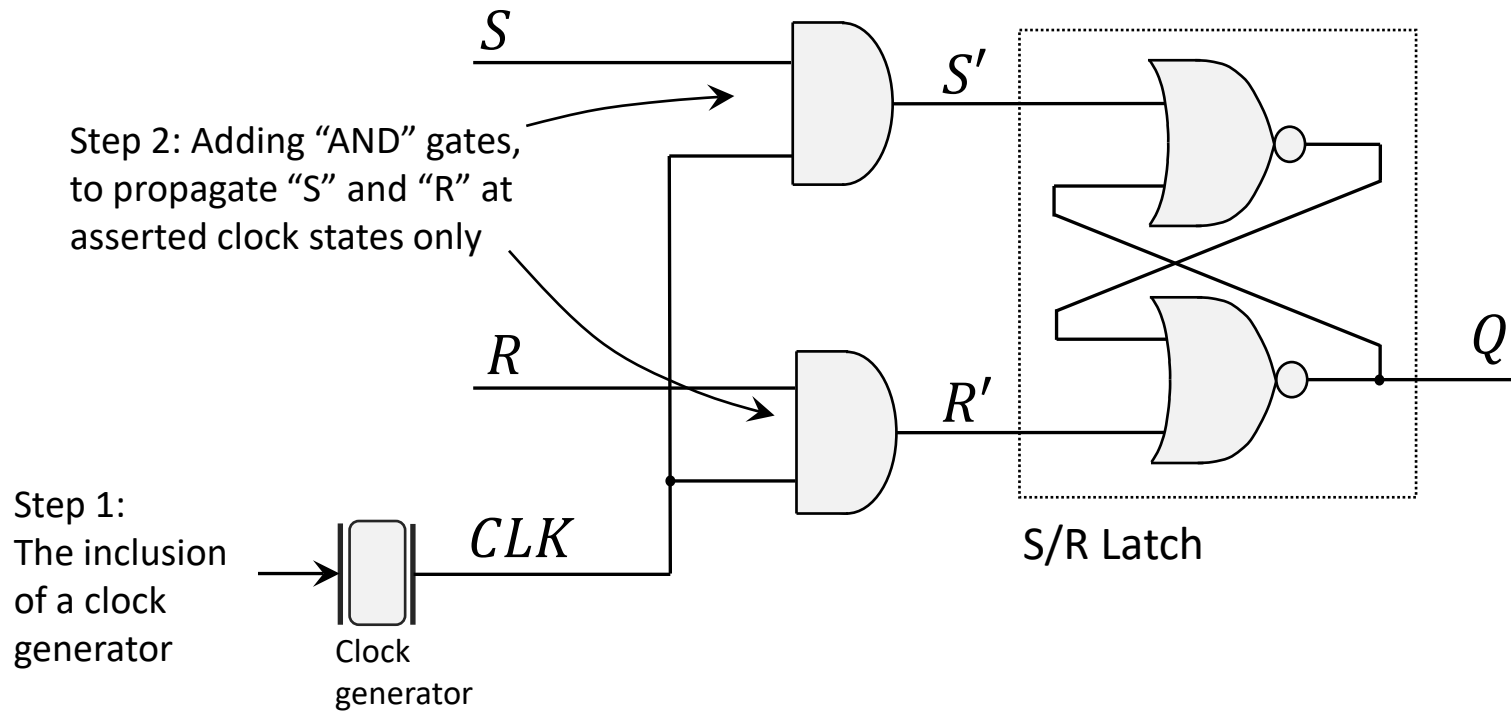
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



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

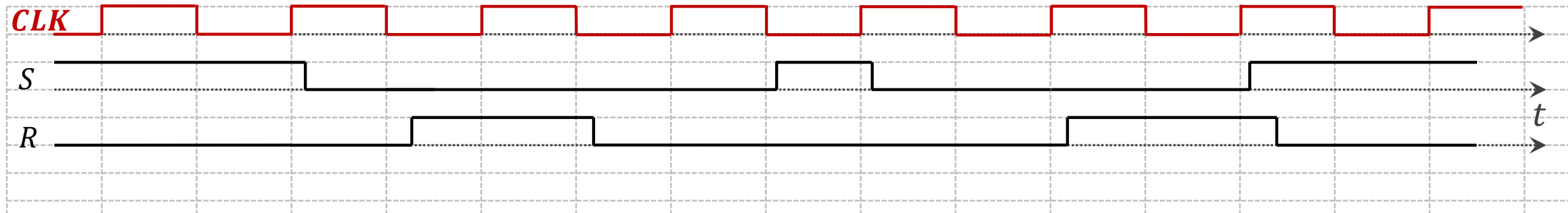


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

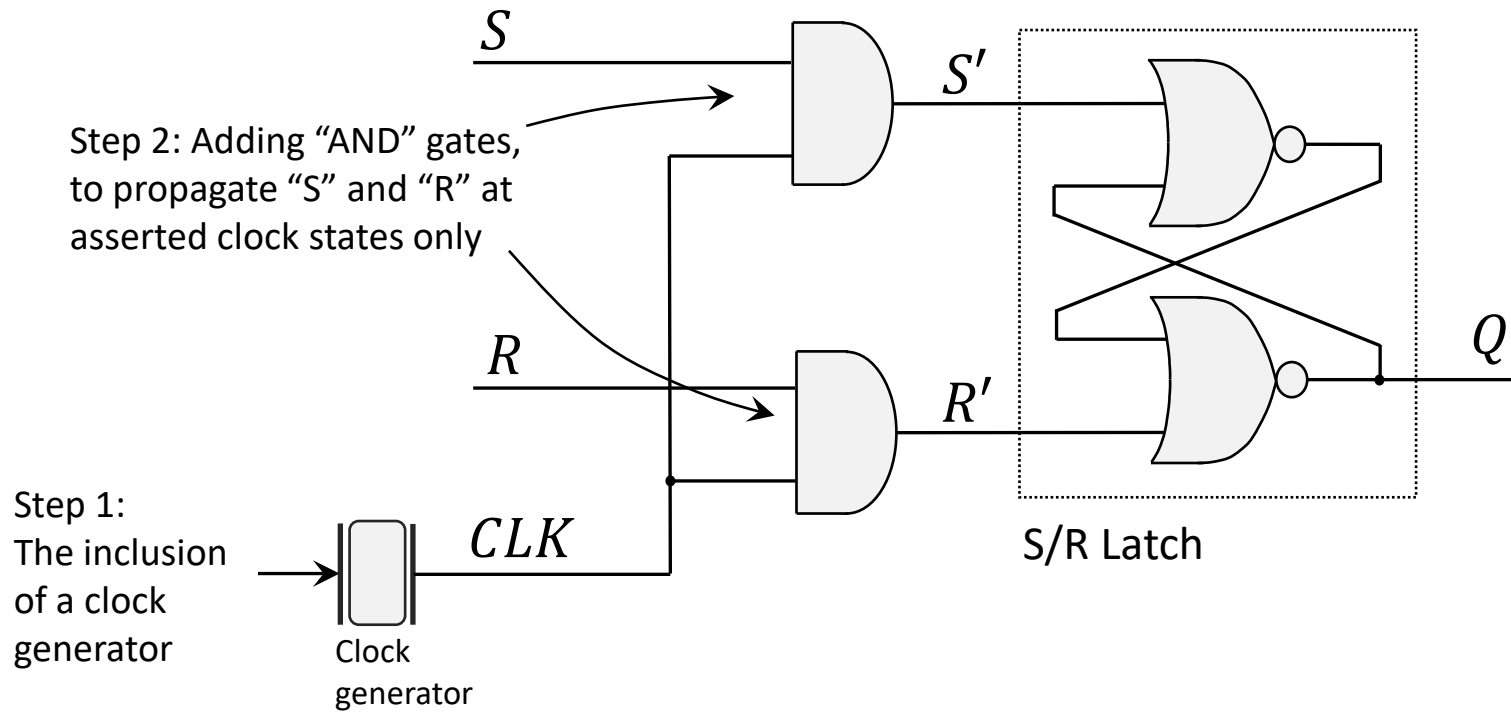


Observation:

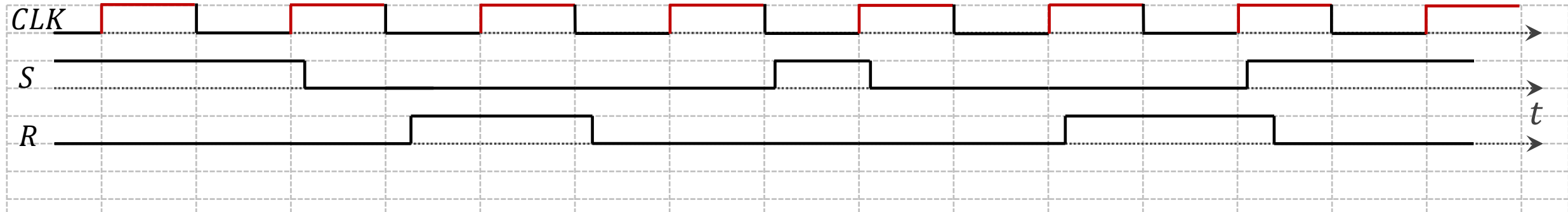
Inputs "S" and "R" make effect on "Q"
only when clock is asserted



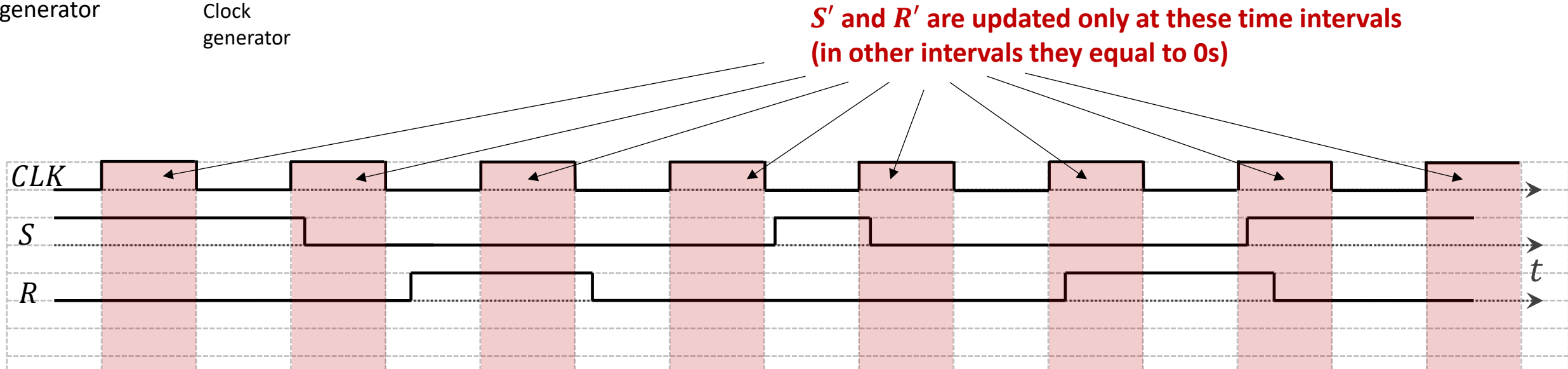
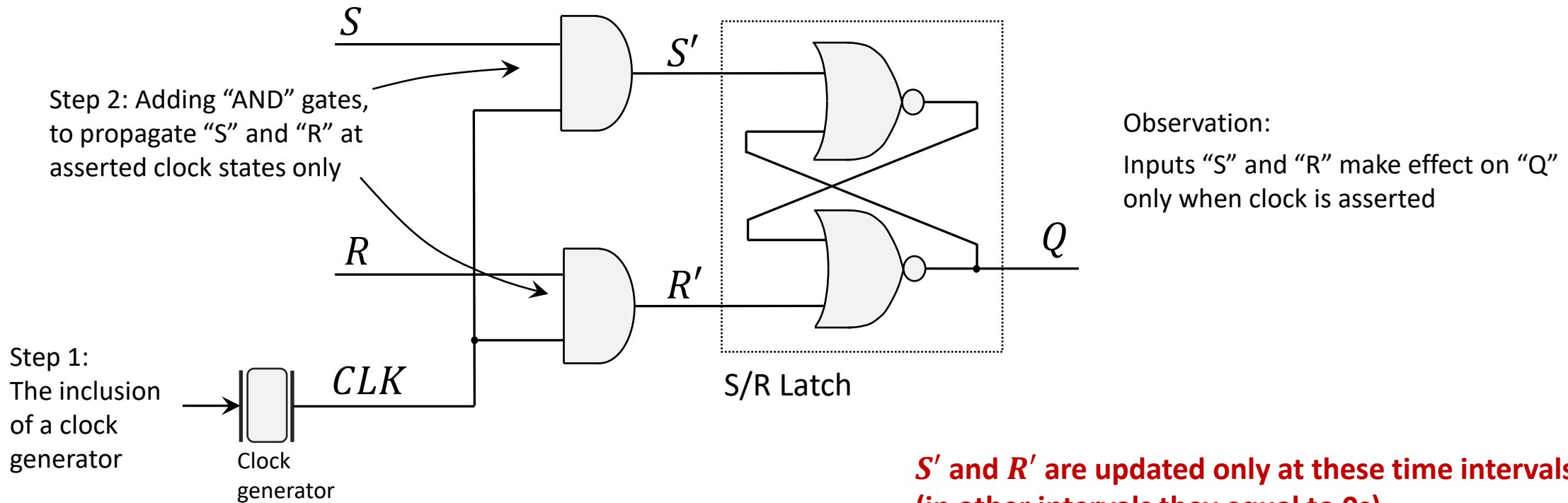
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



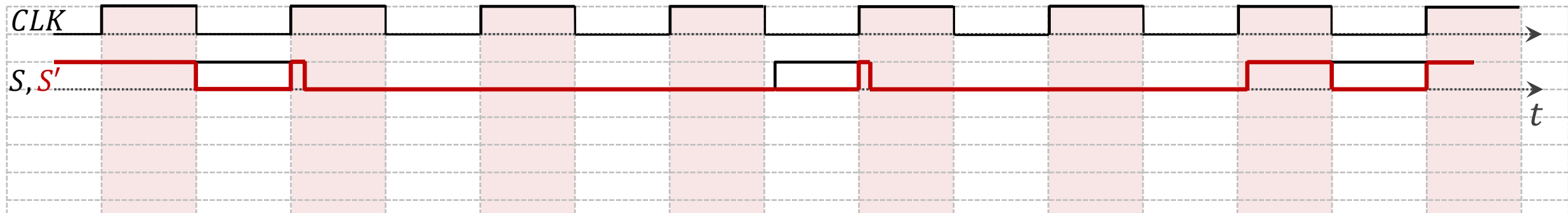
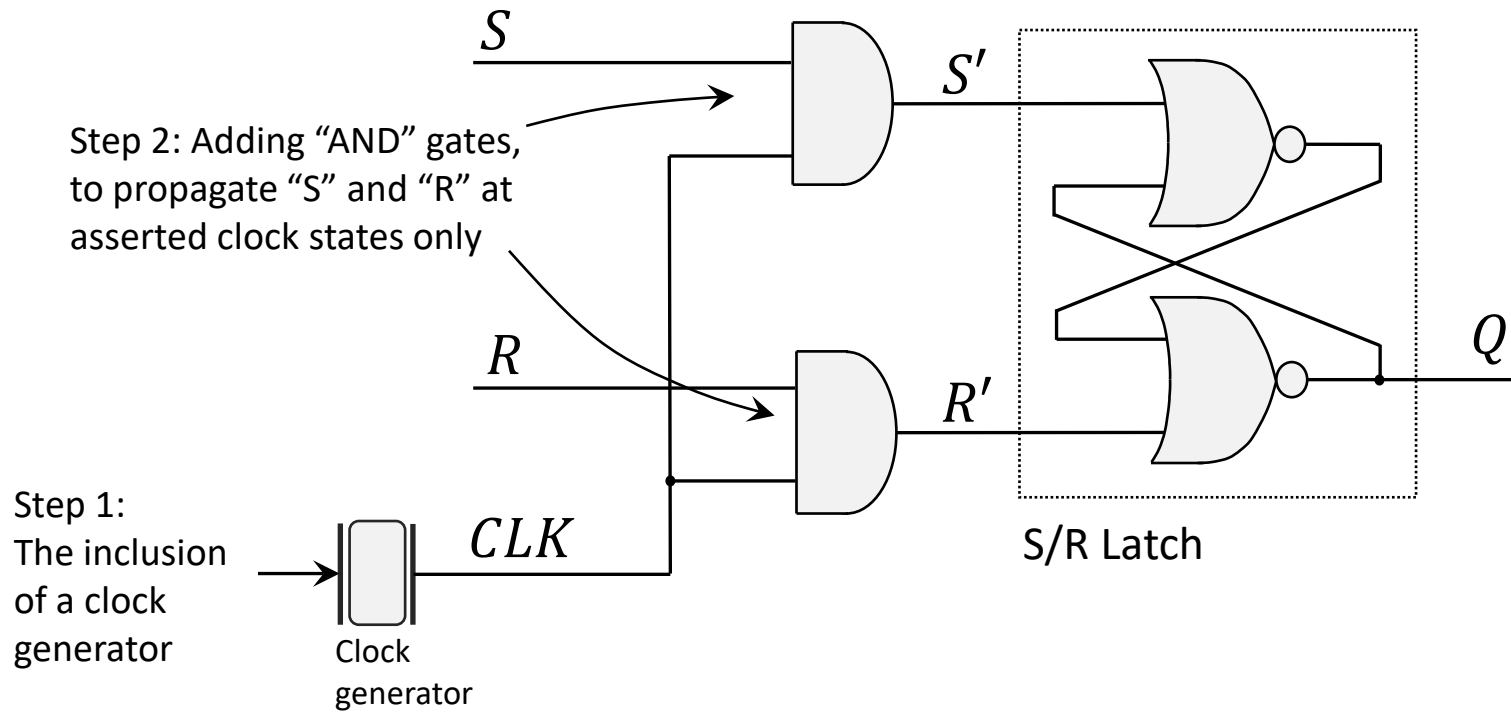
Observation:
Inputs "S" and "R" make effect on "Q"
only when clock is asserted



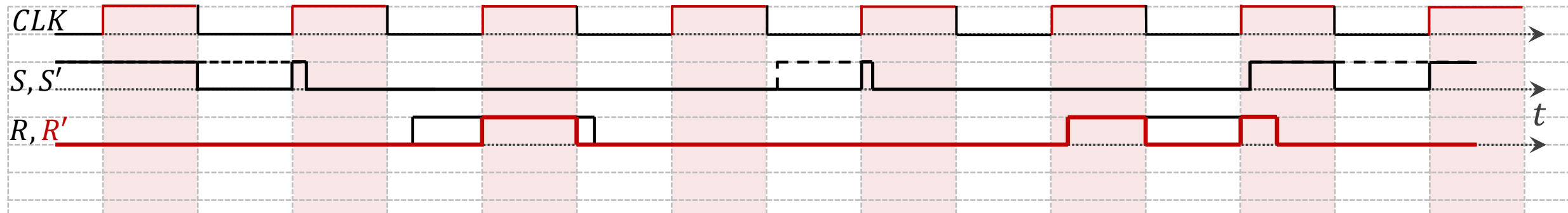
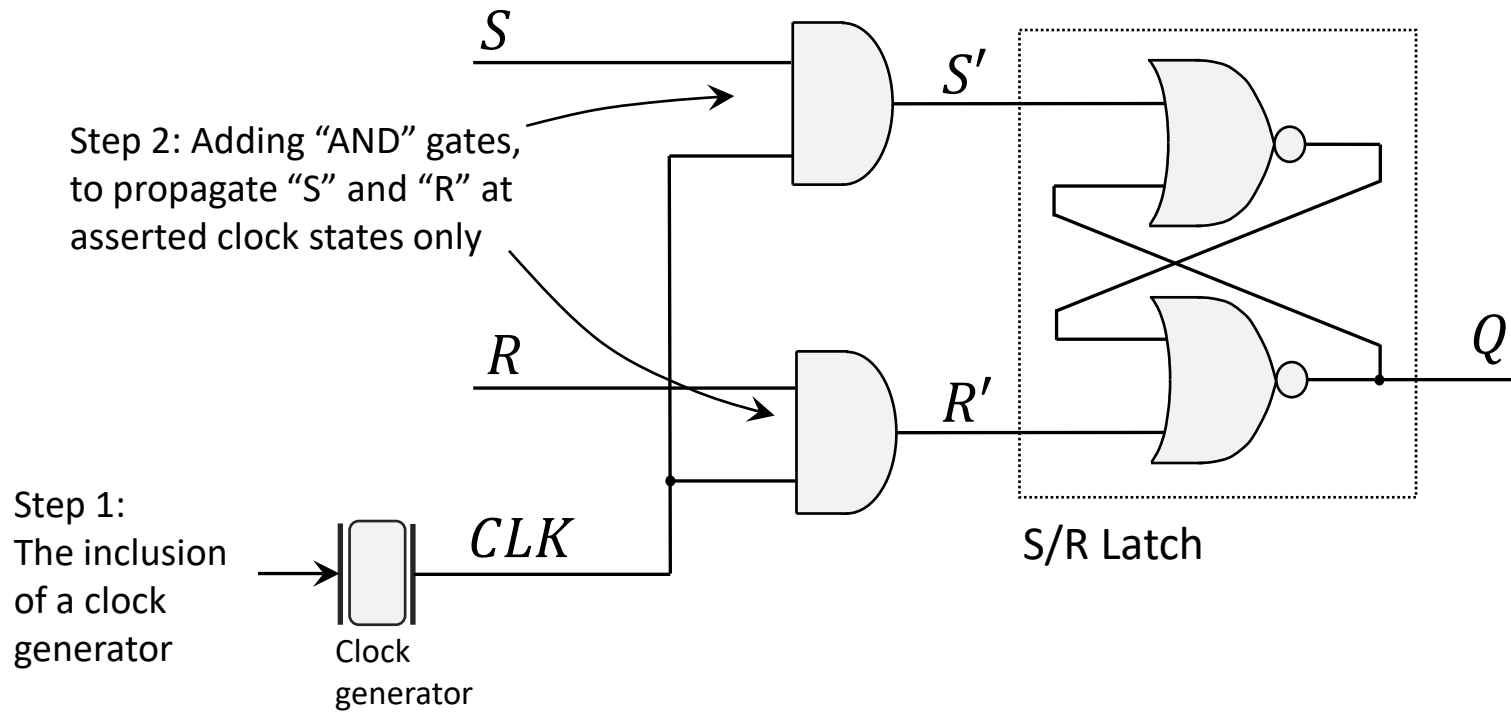
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



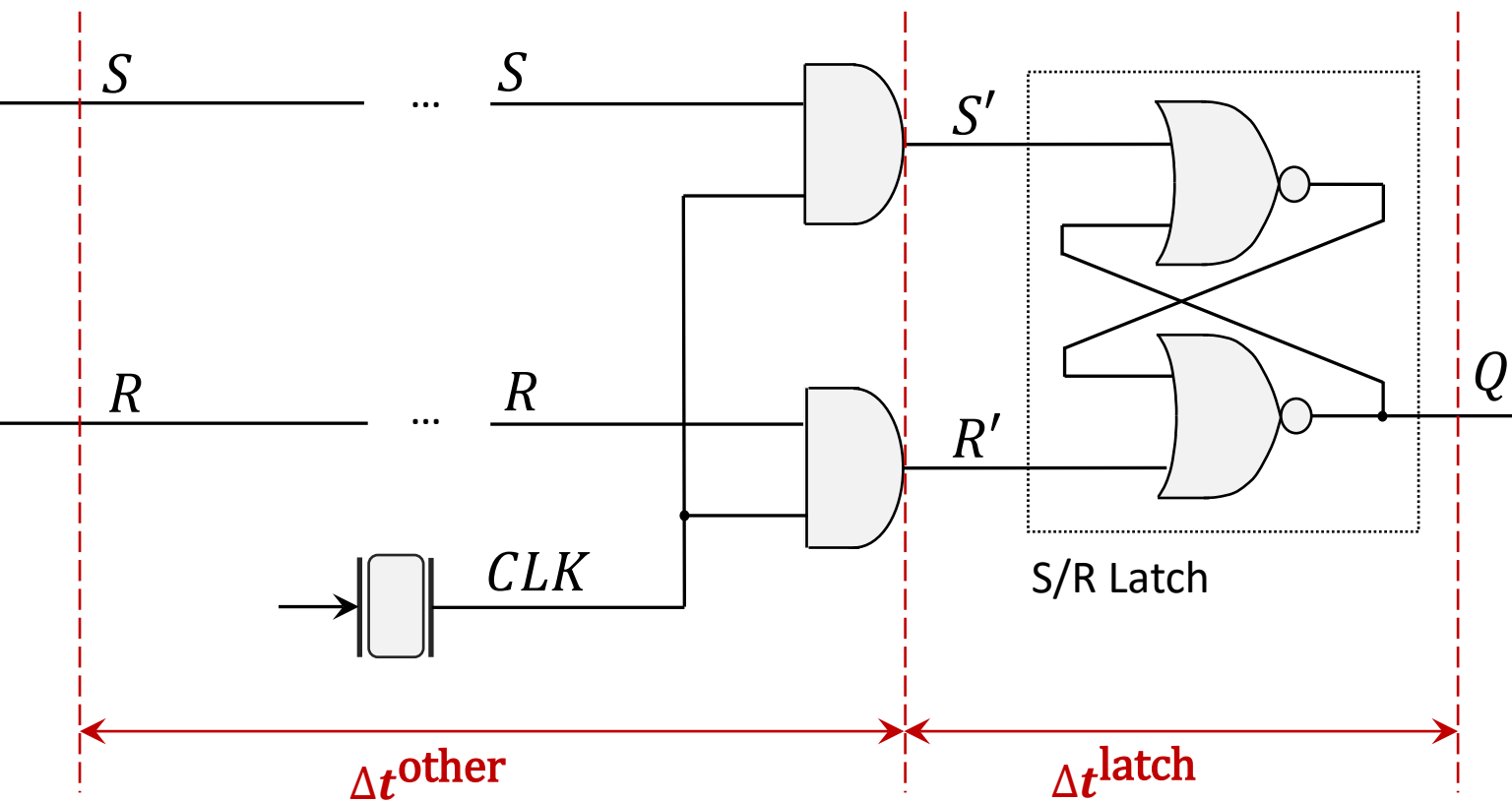
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

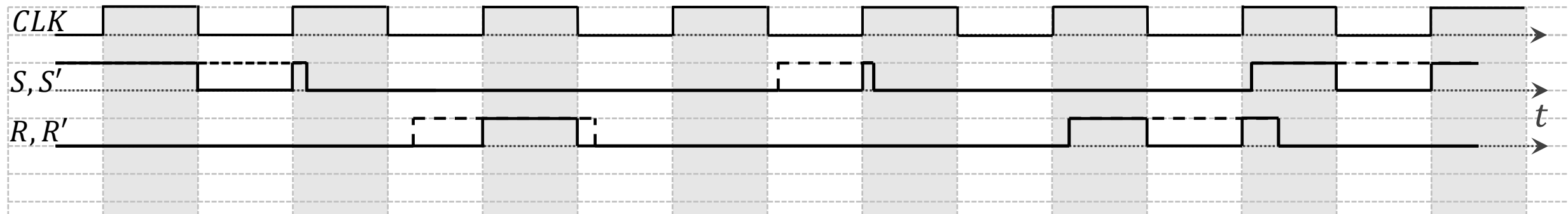


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

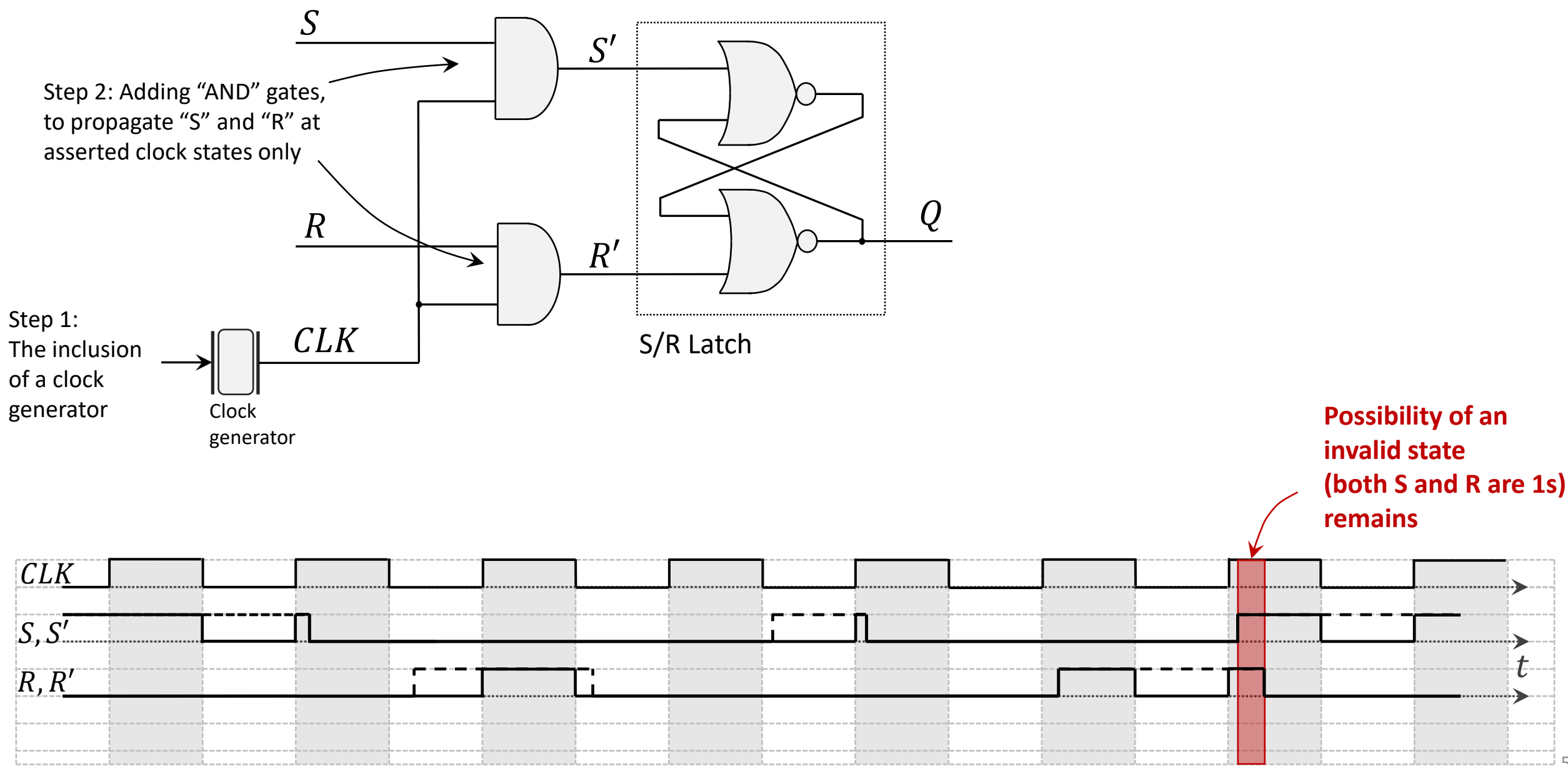


Assumption:

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

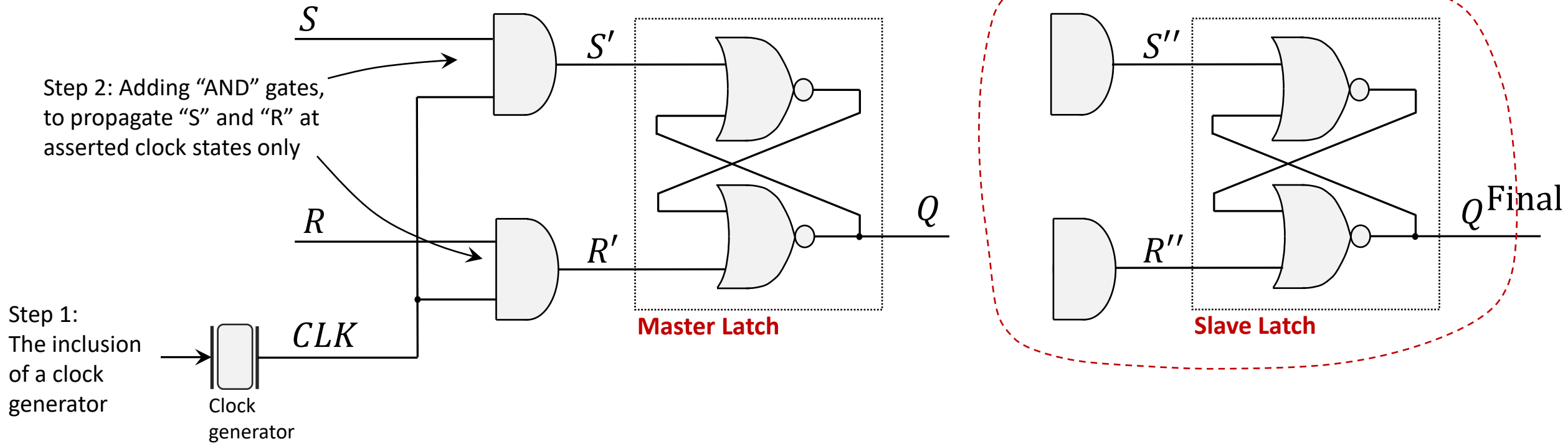


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



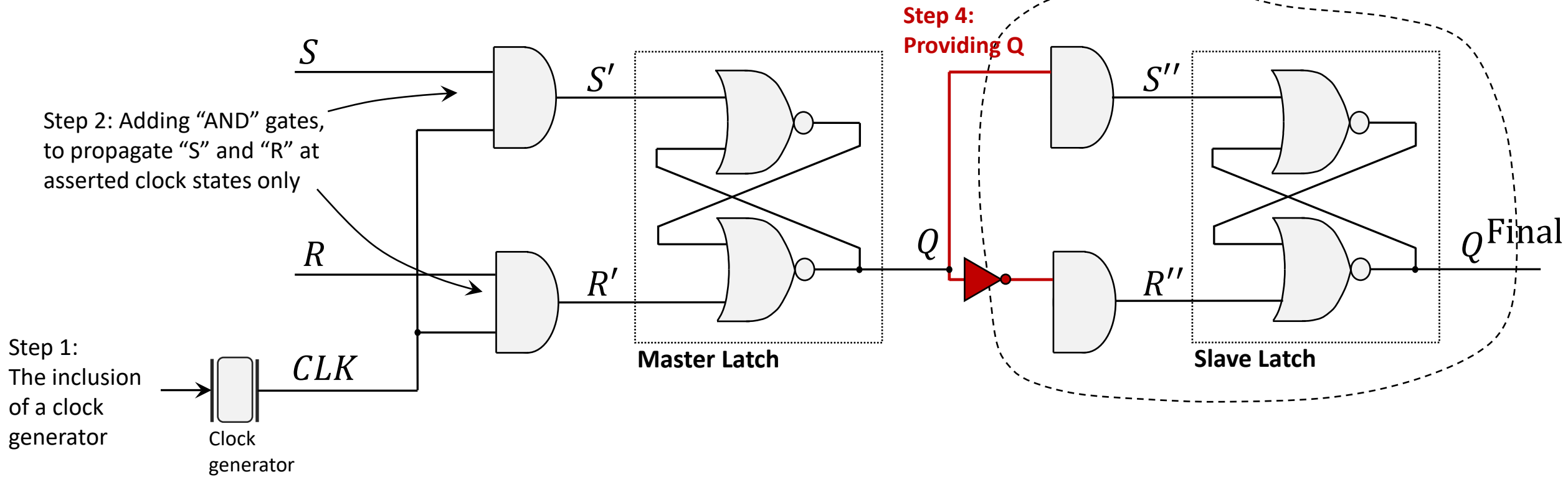
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

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

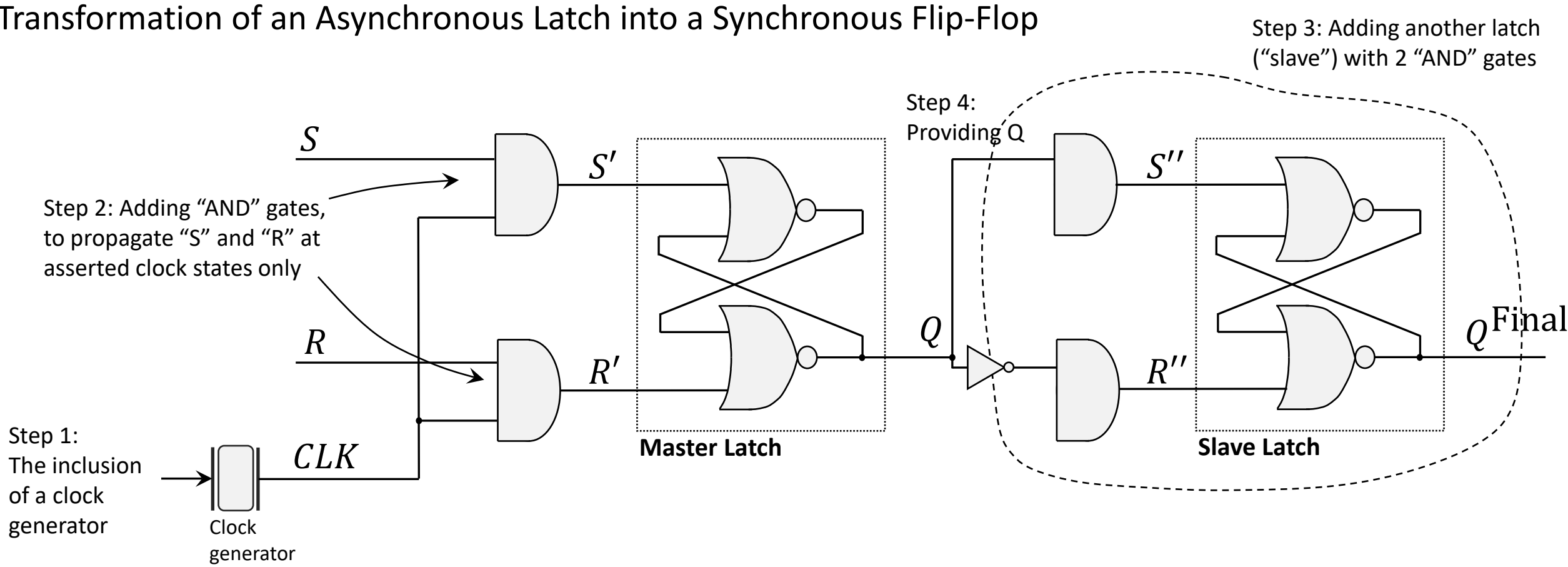


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

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

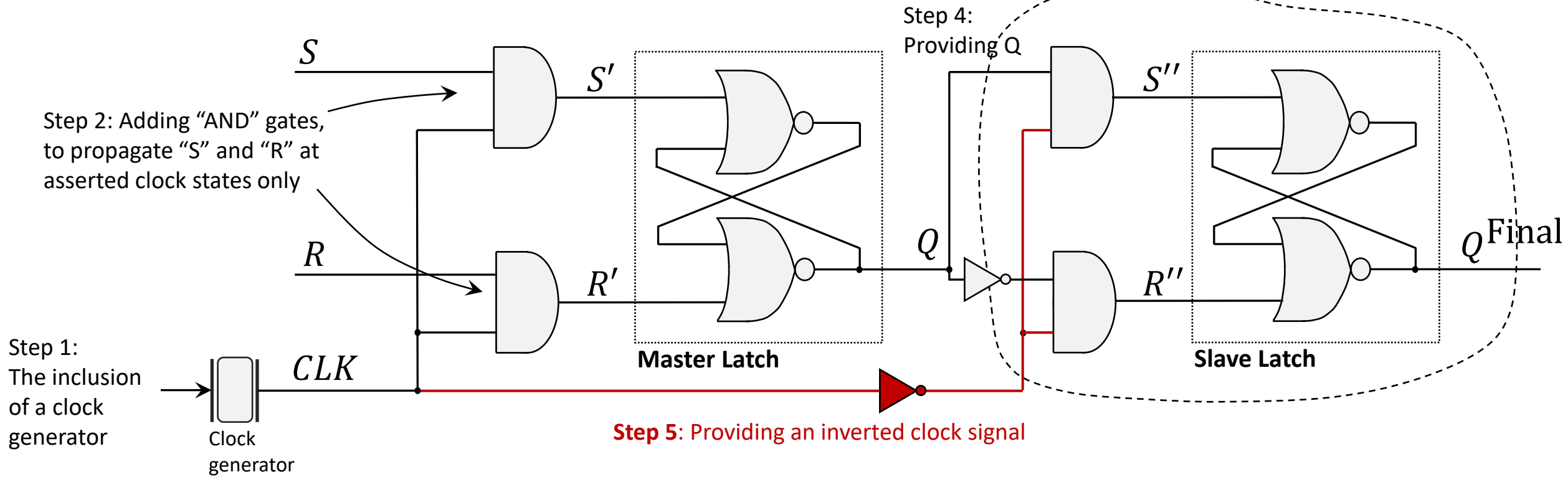


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

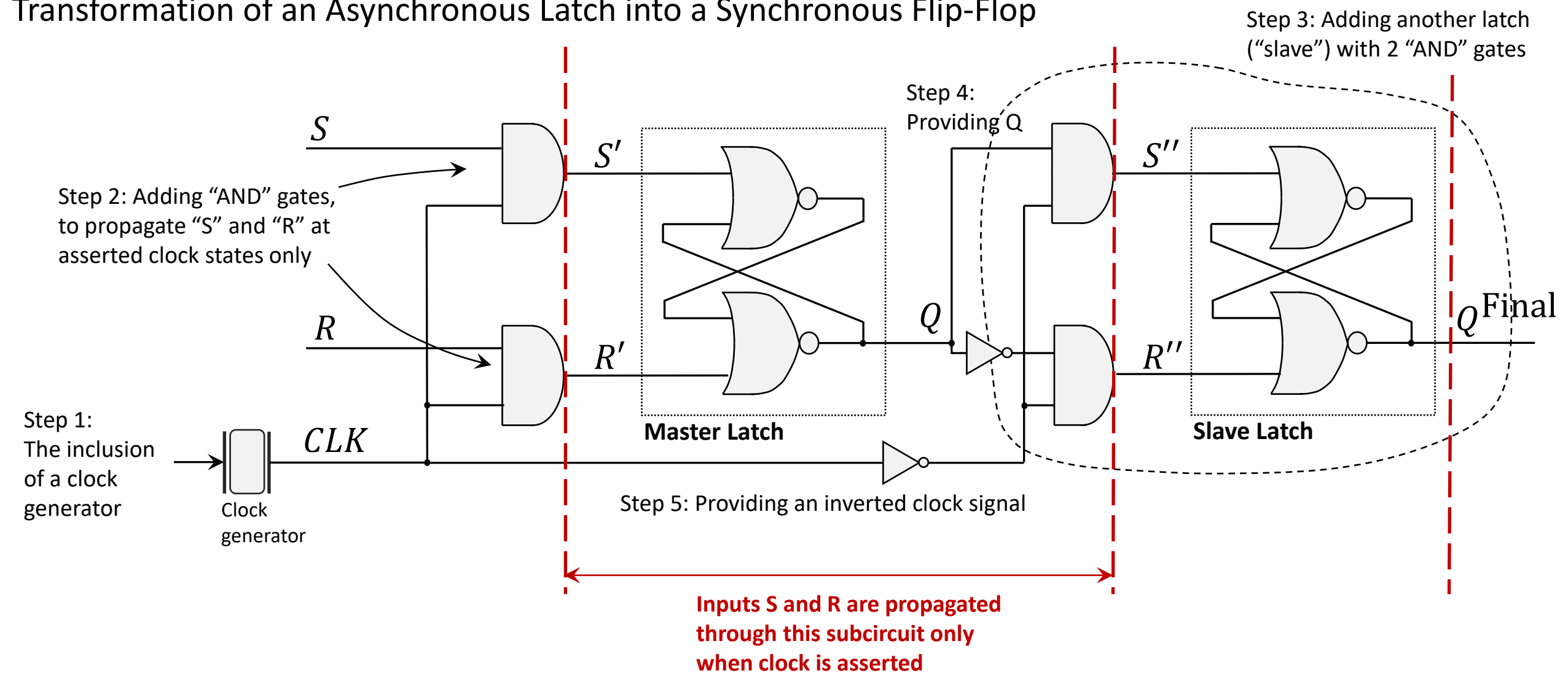


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

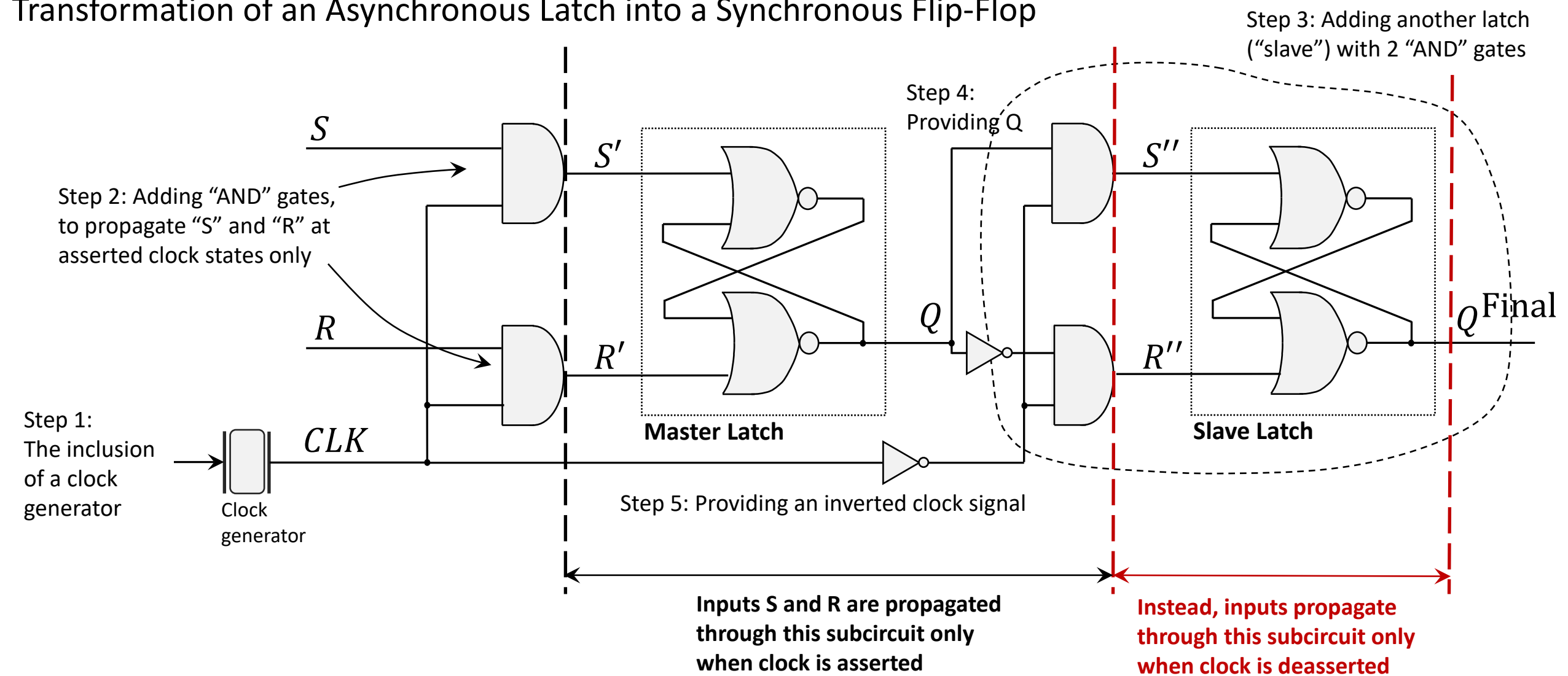
Step 3: Adding another latch ("slave") with 2 "AND" gates



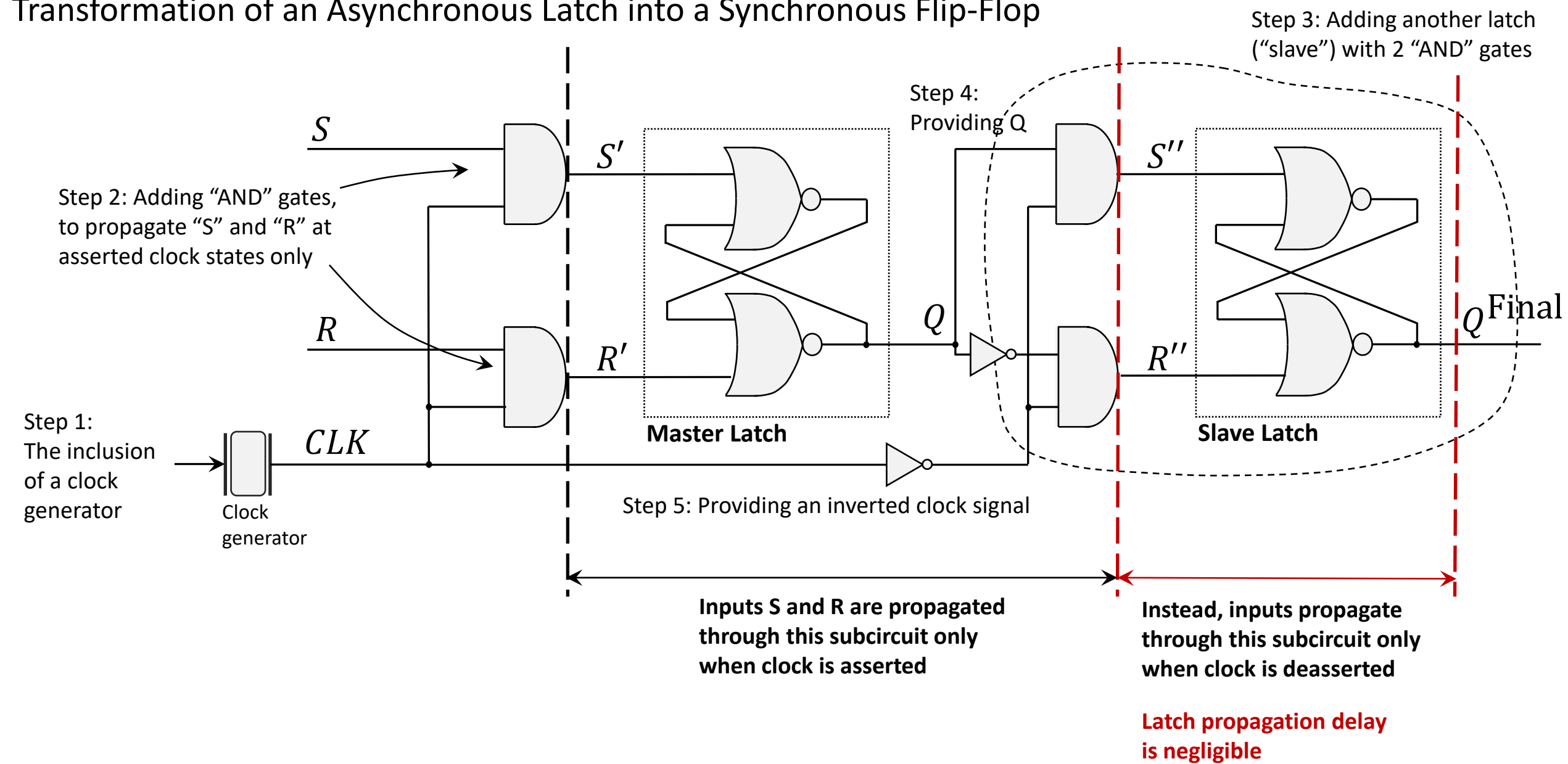
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



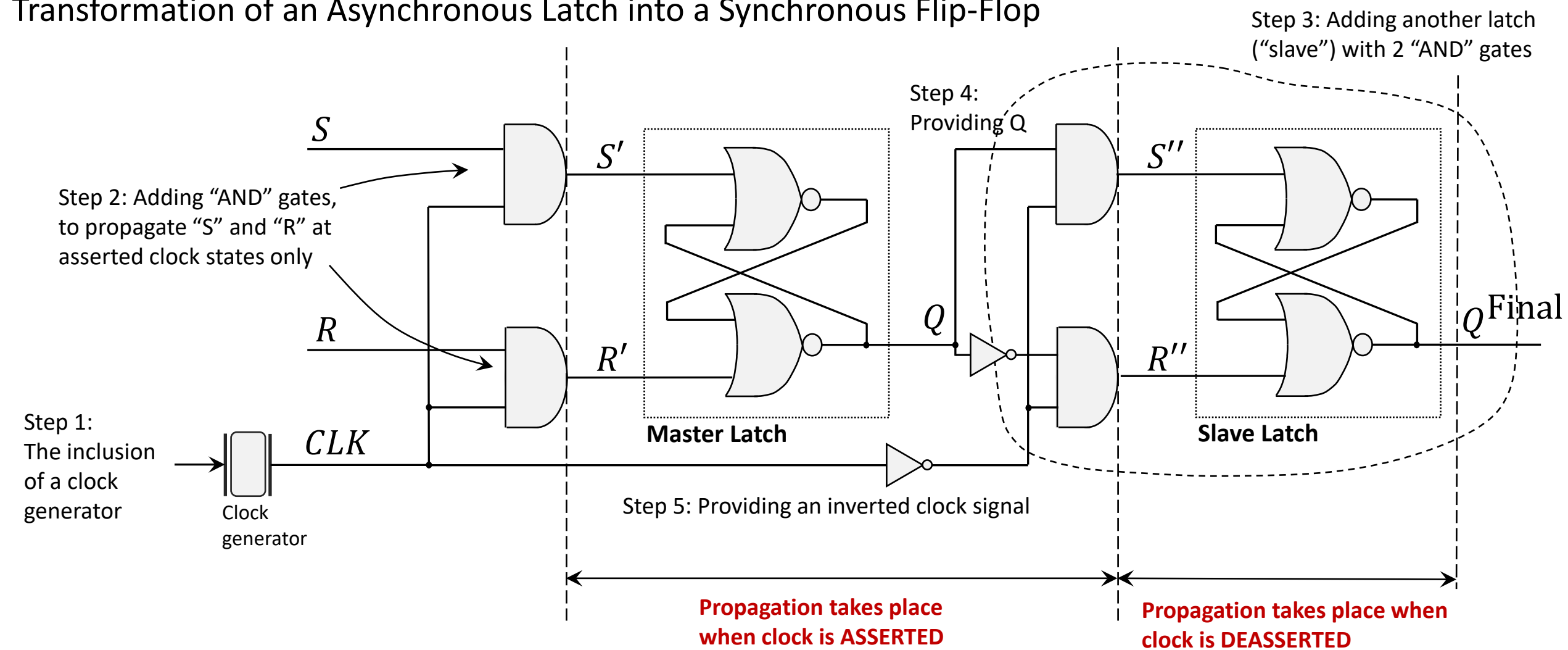
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



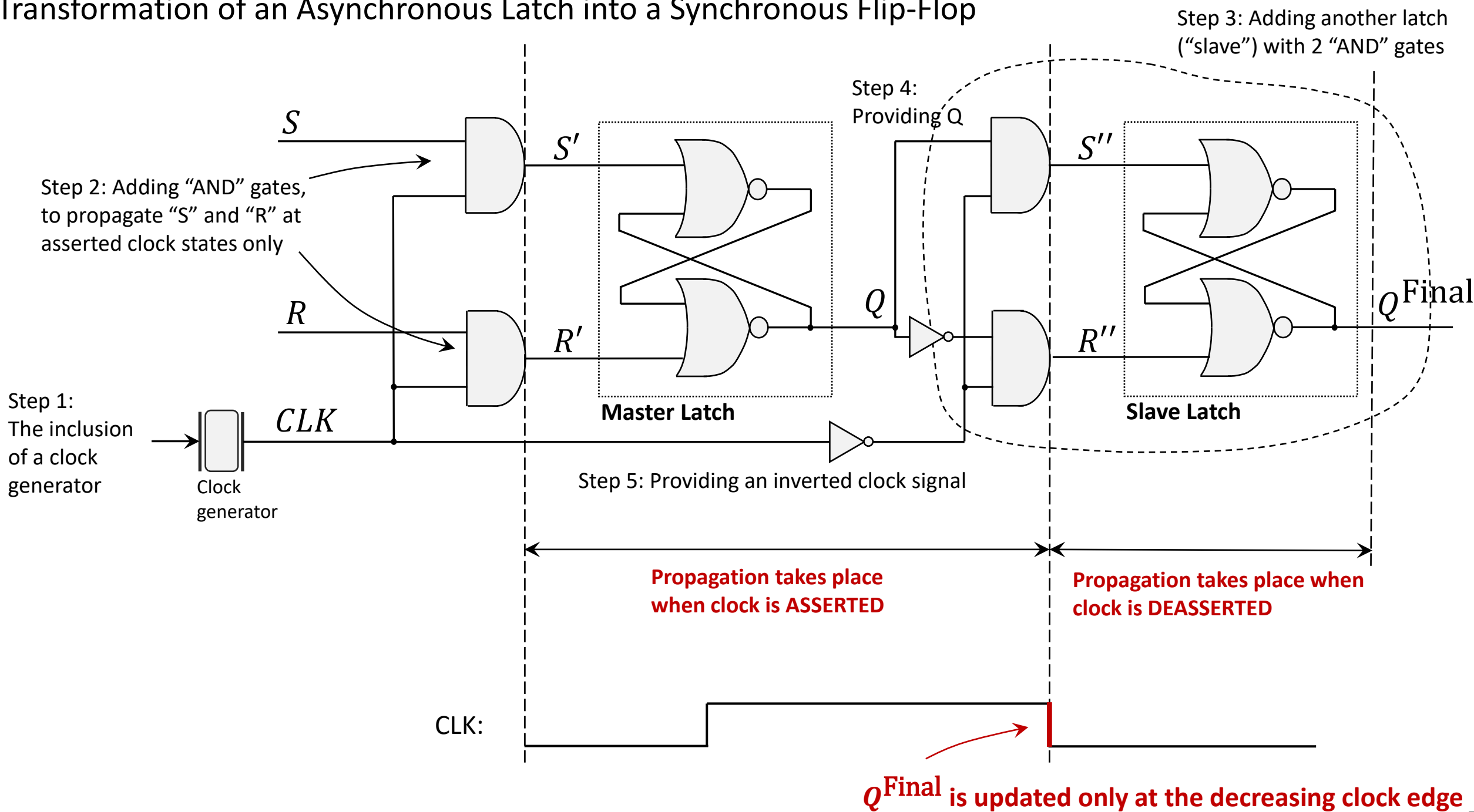
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



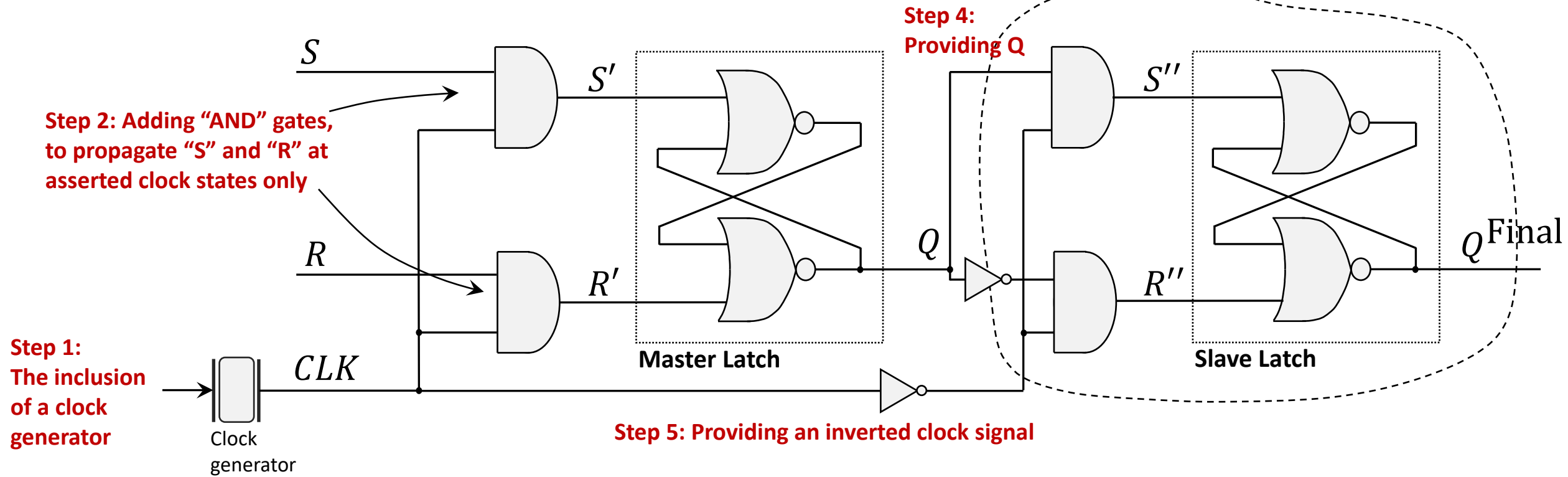
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



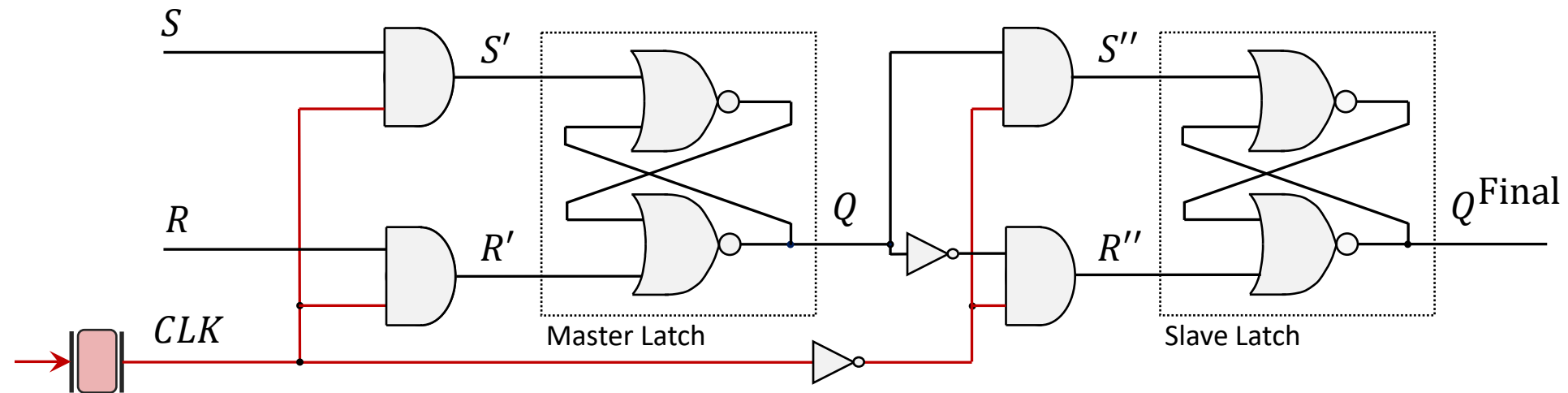
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

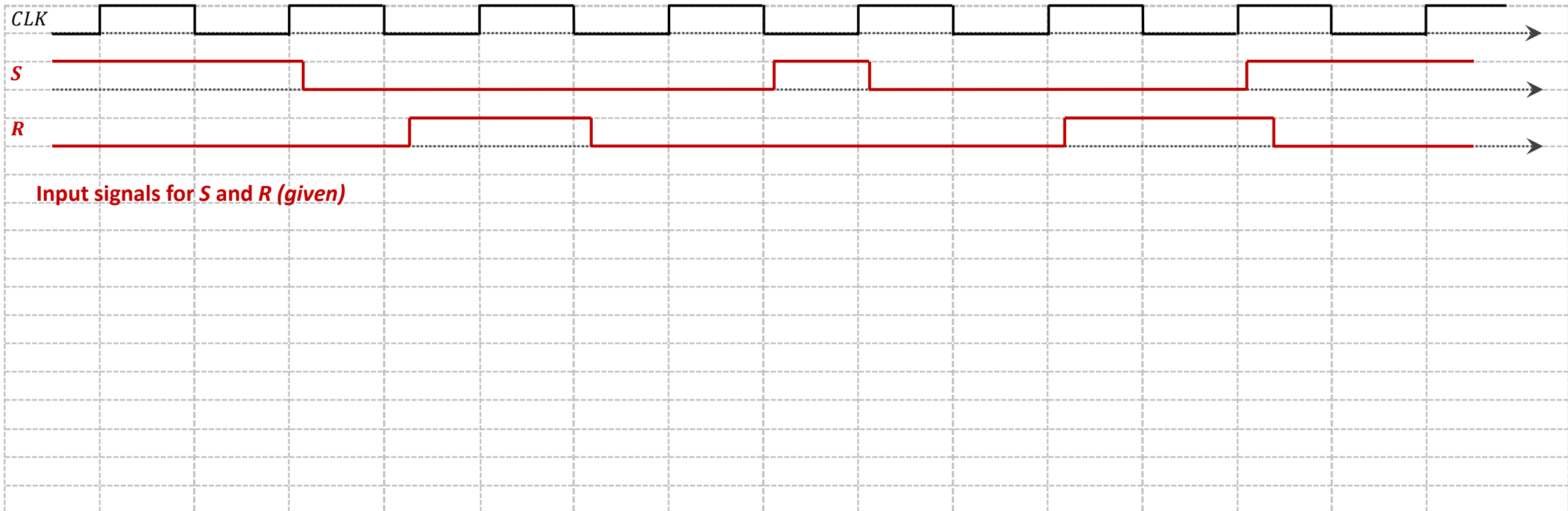
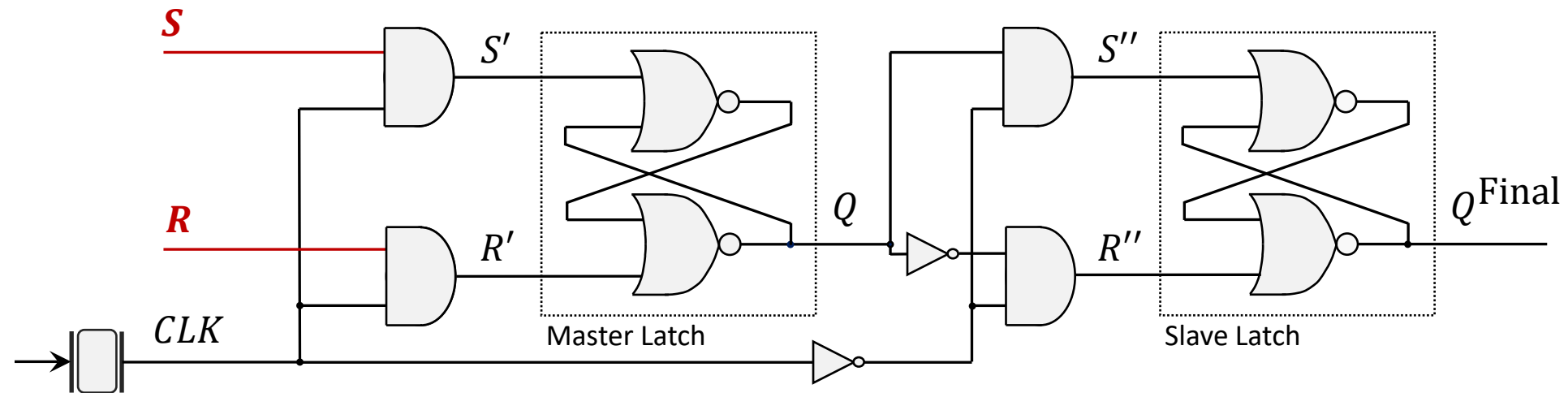


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

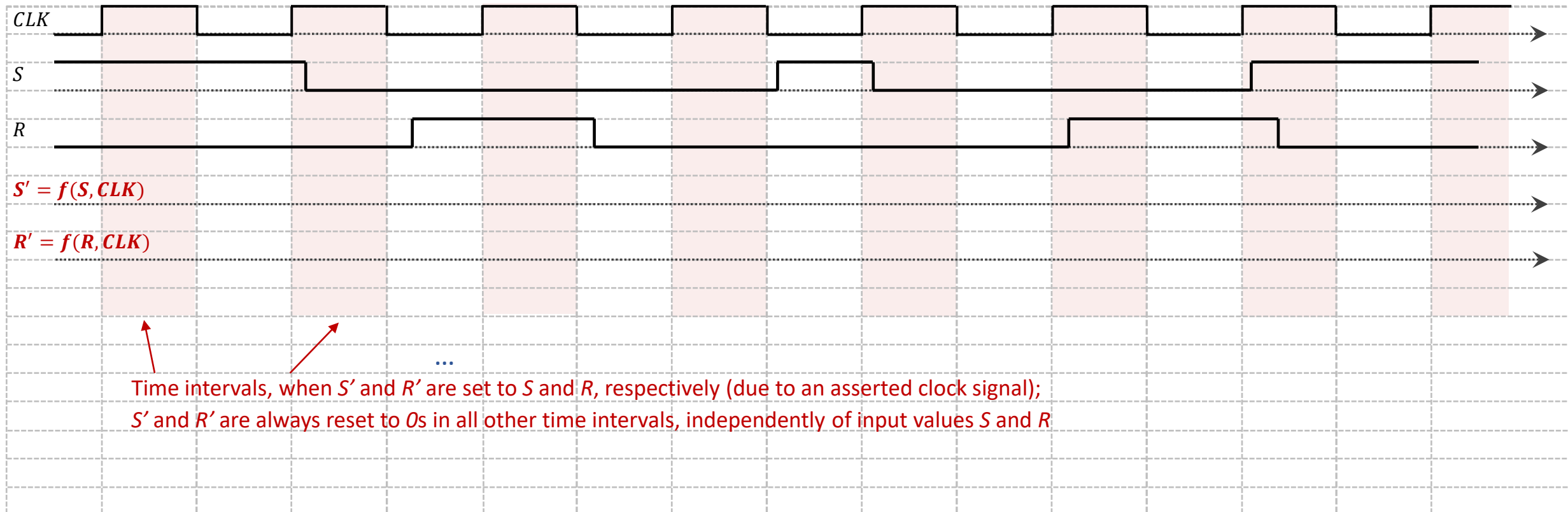
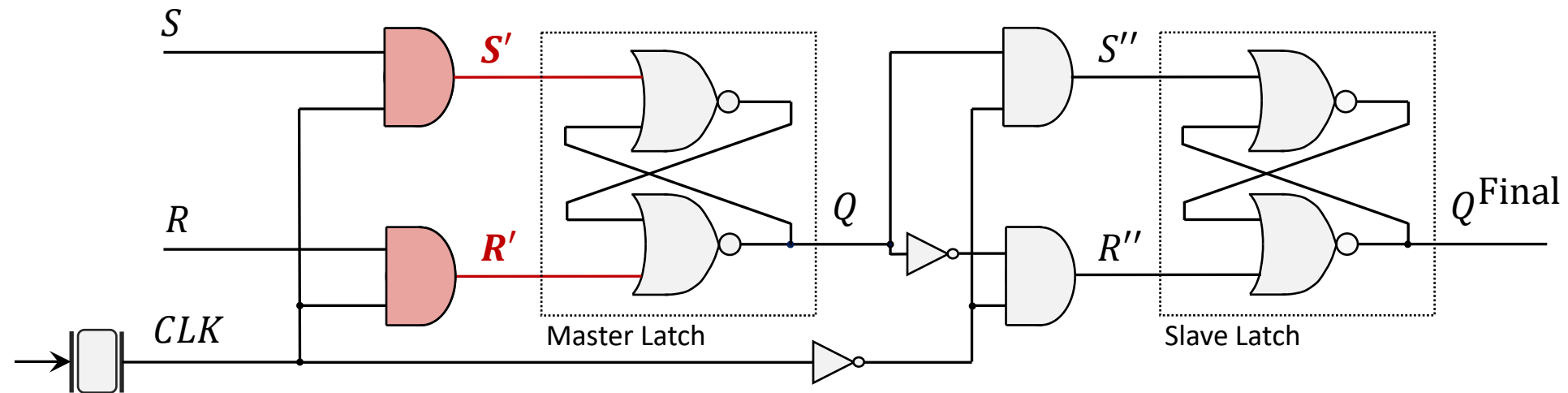


A clock signal, which governs the entire activity (that is the propagation of electrical signals) in our circuit;
A clock periodicity is decided by a computer architect, accounting the expected propagation delays

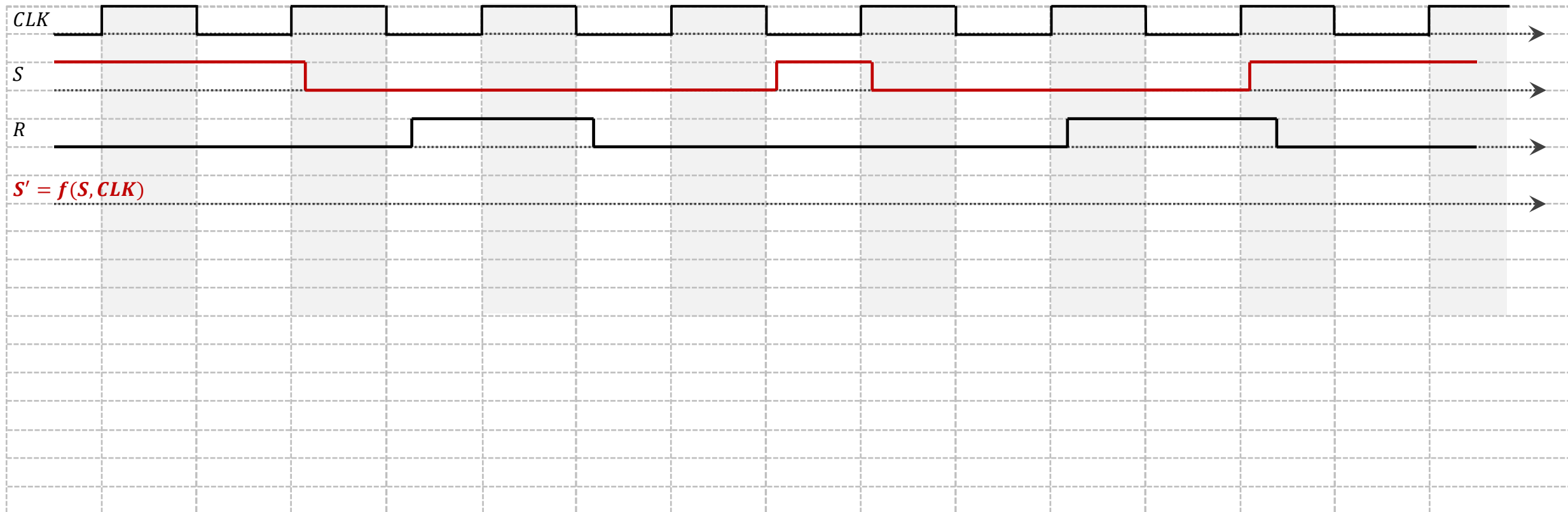
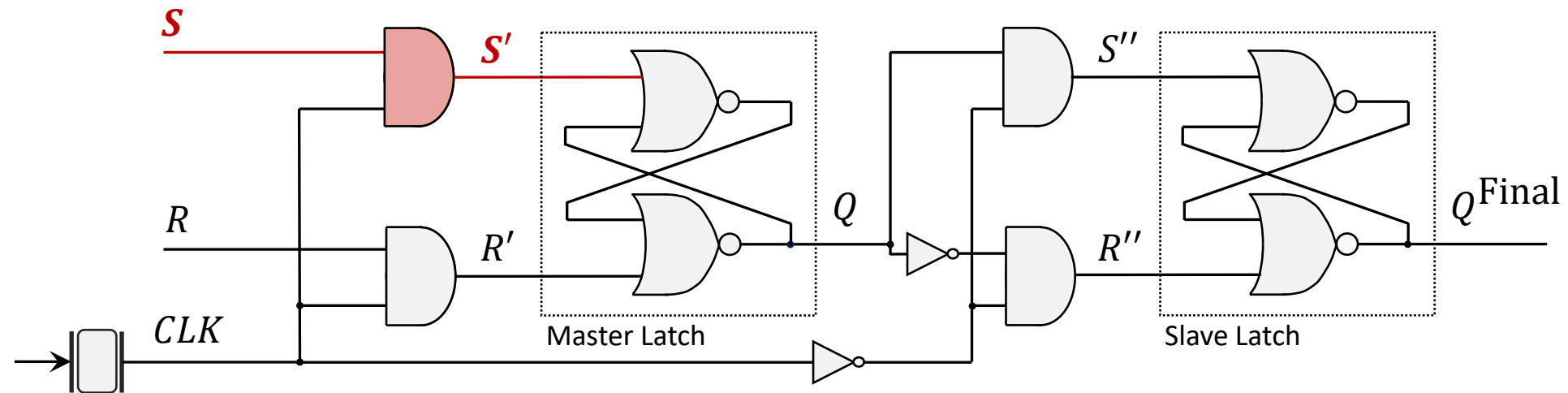
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



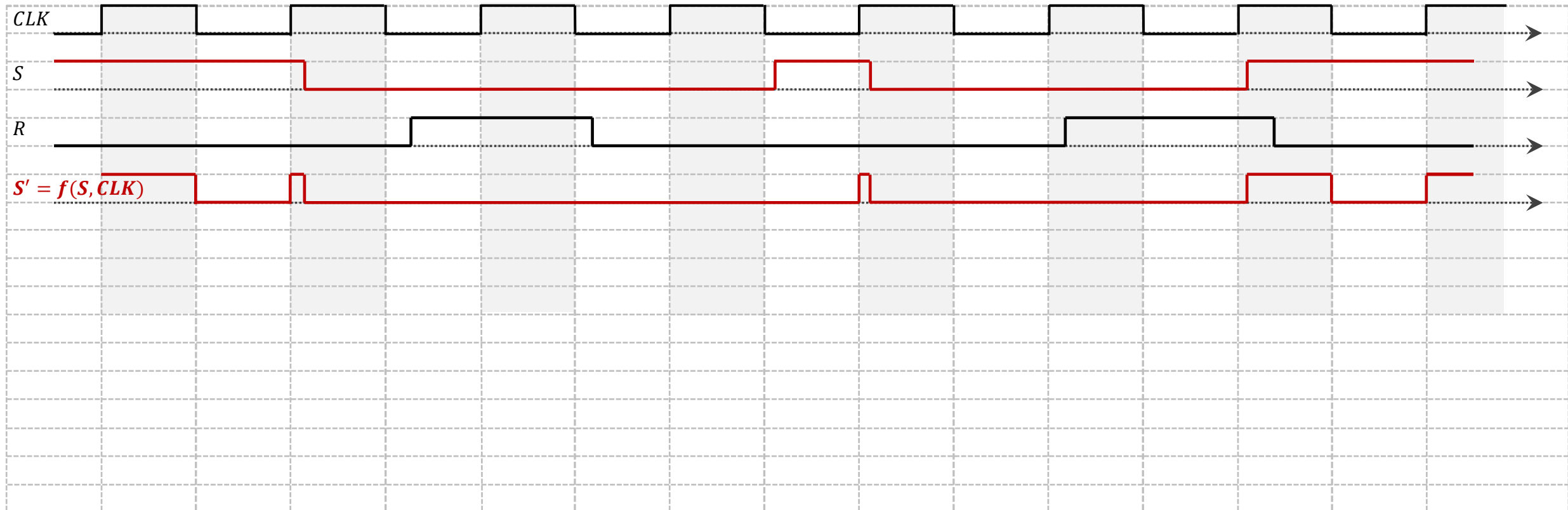
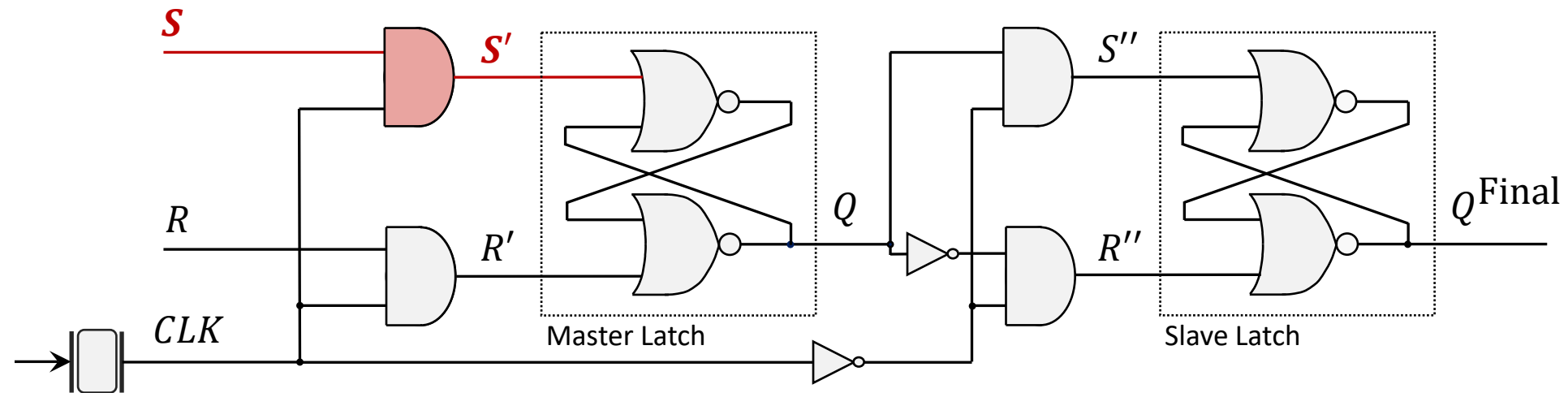
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



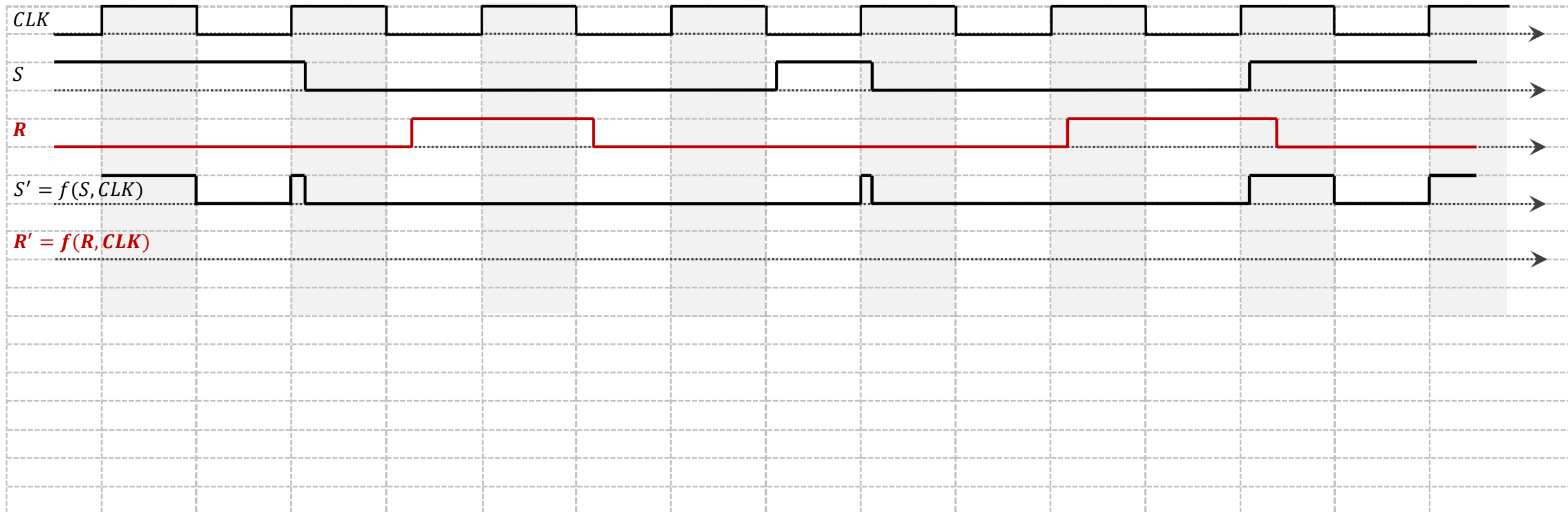
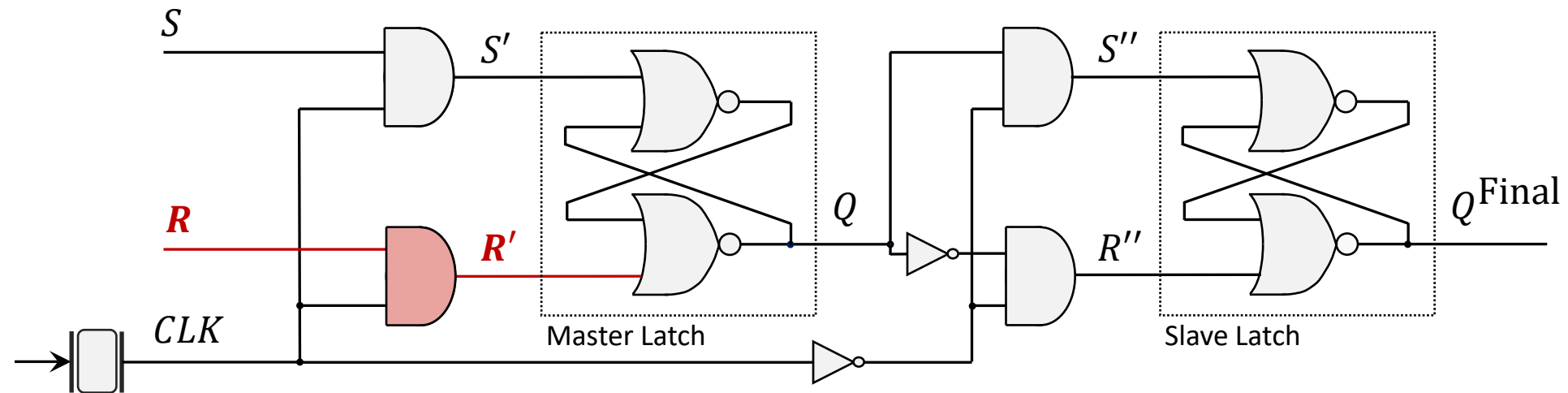
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



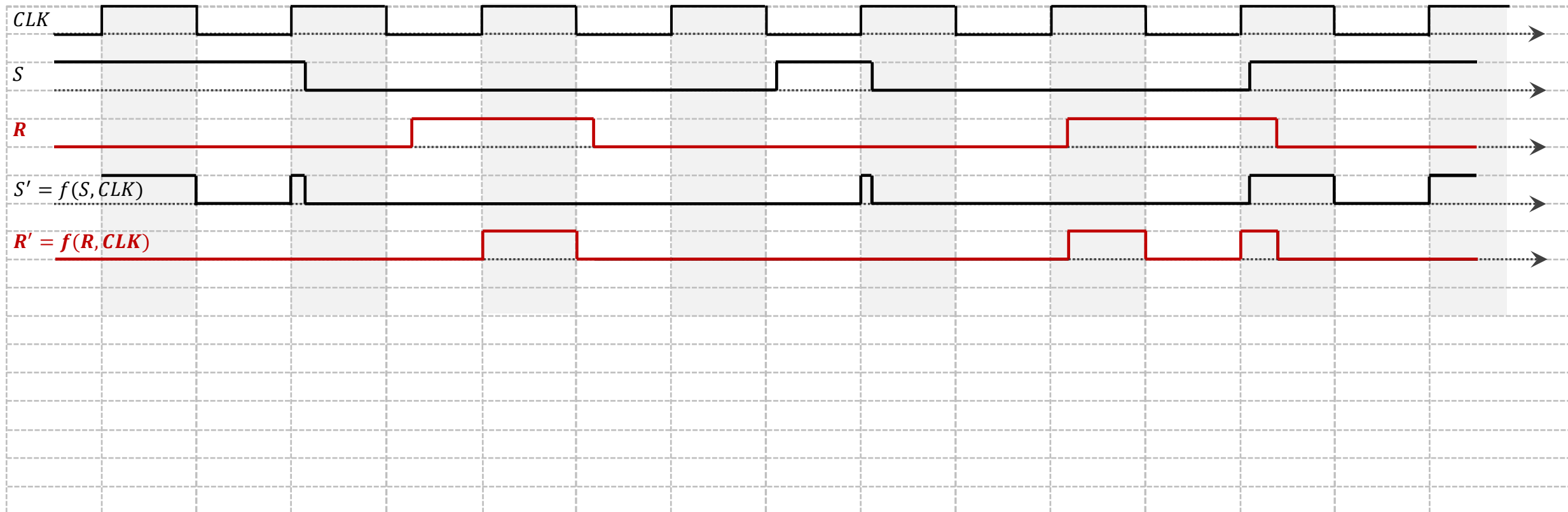
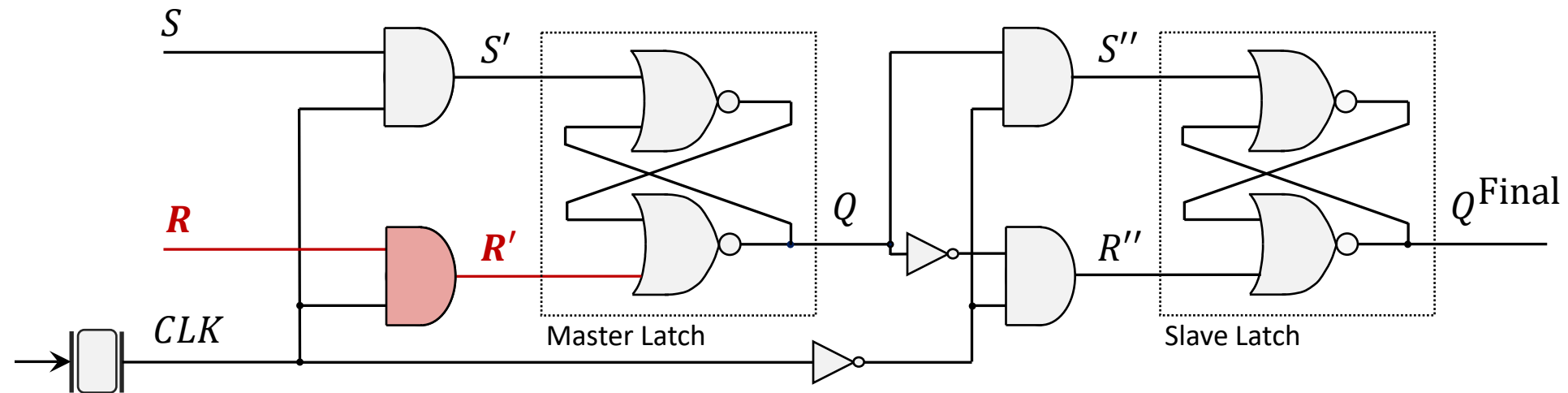
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



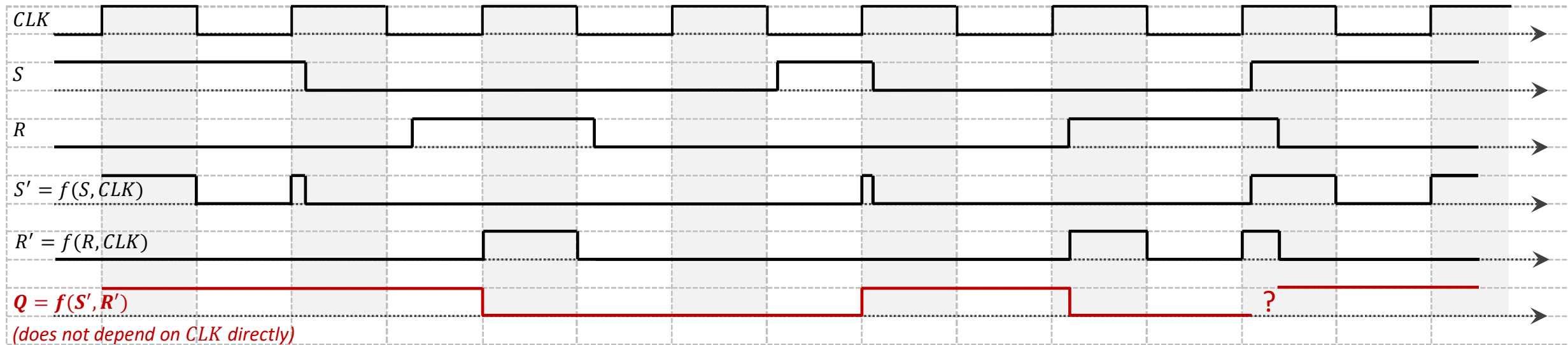
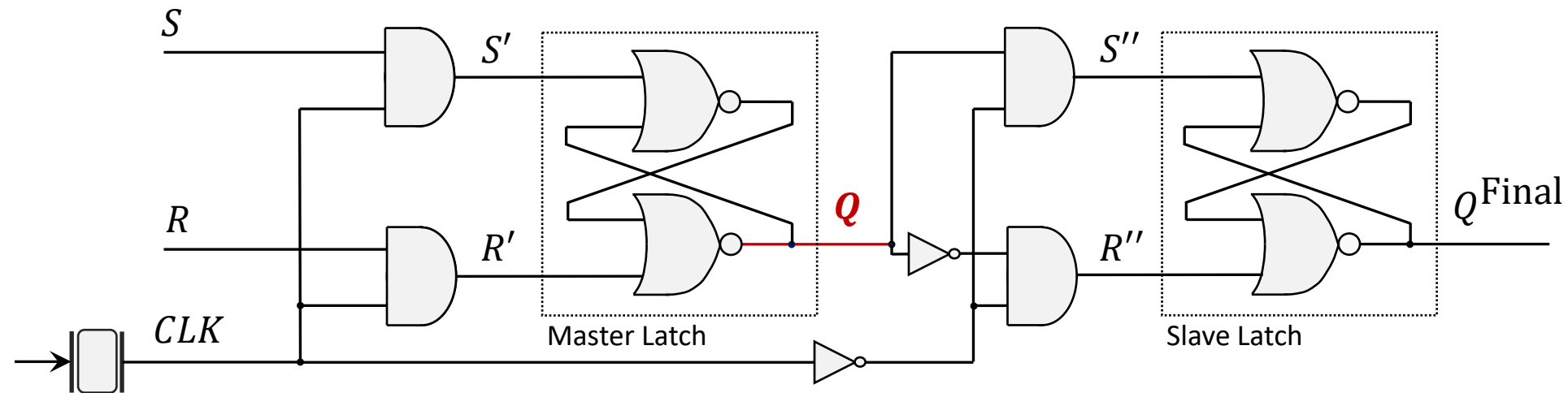
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

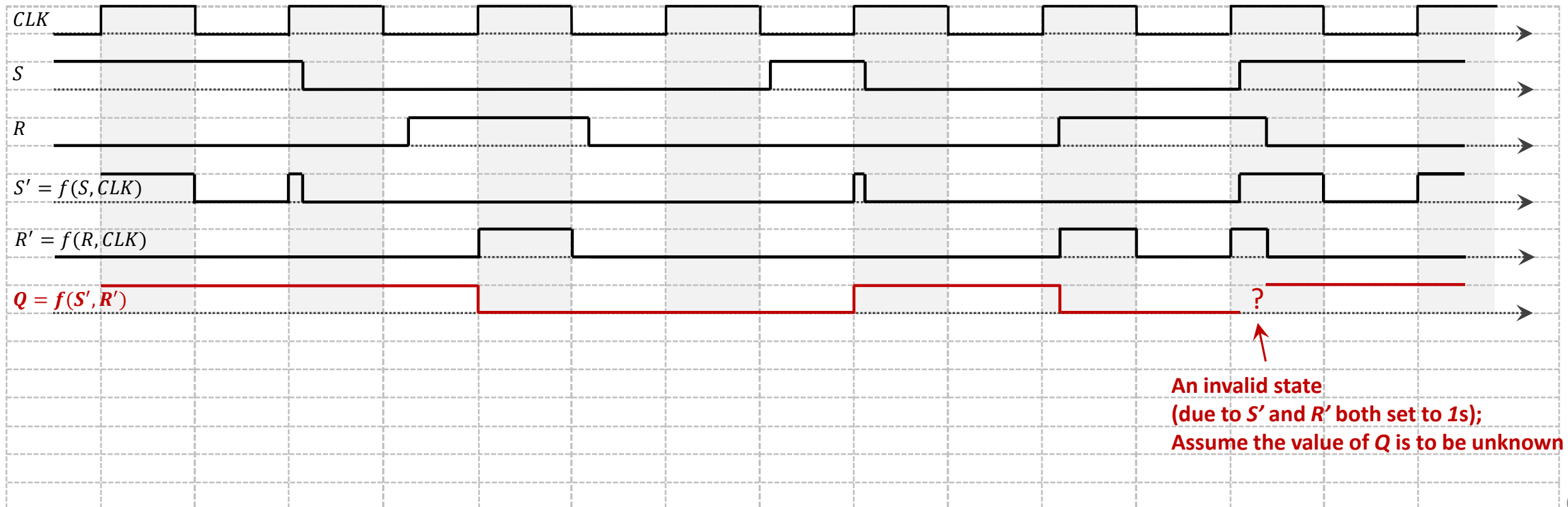
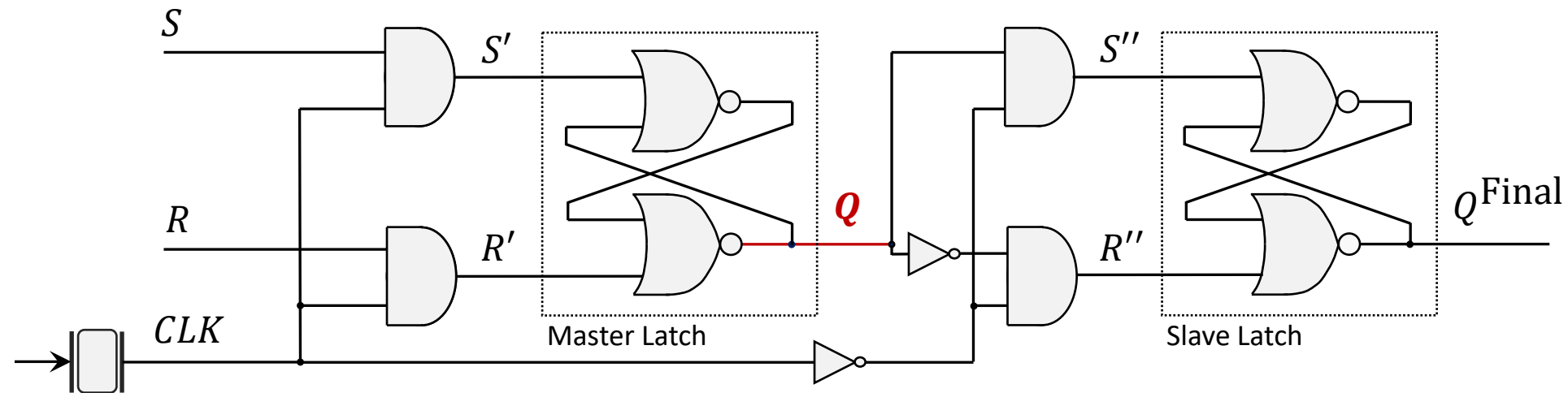


Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

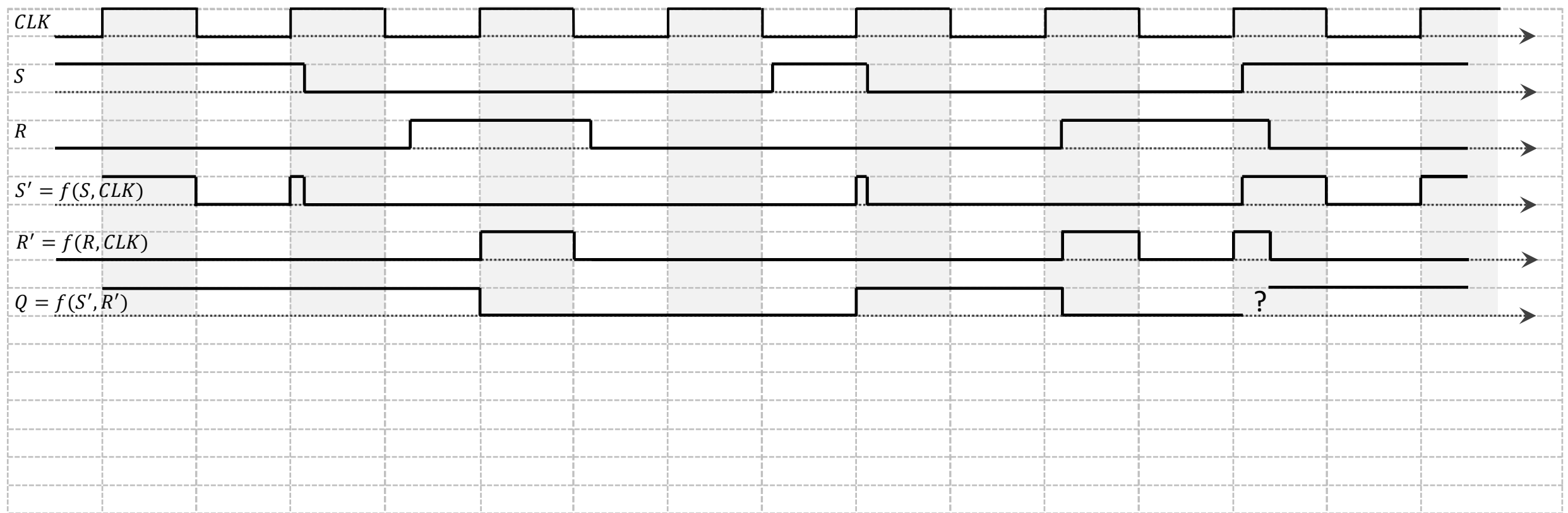
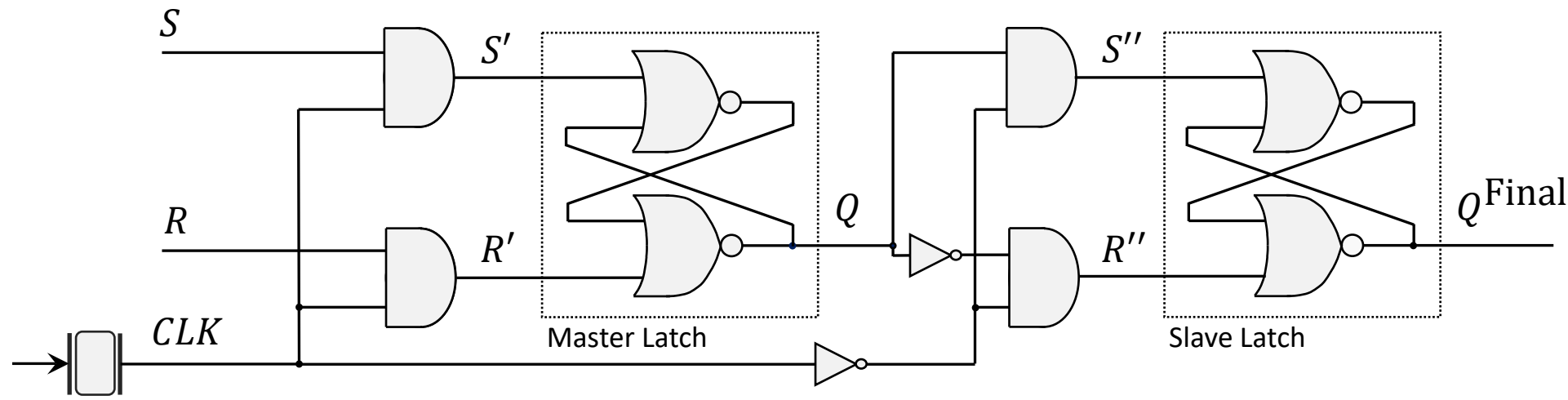


Recap: Whenever R' and S' go to 0s, Q retains its value unchanged (the property of a latch)

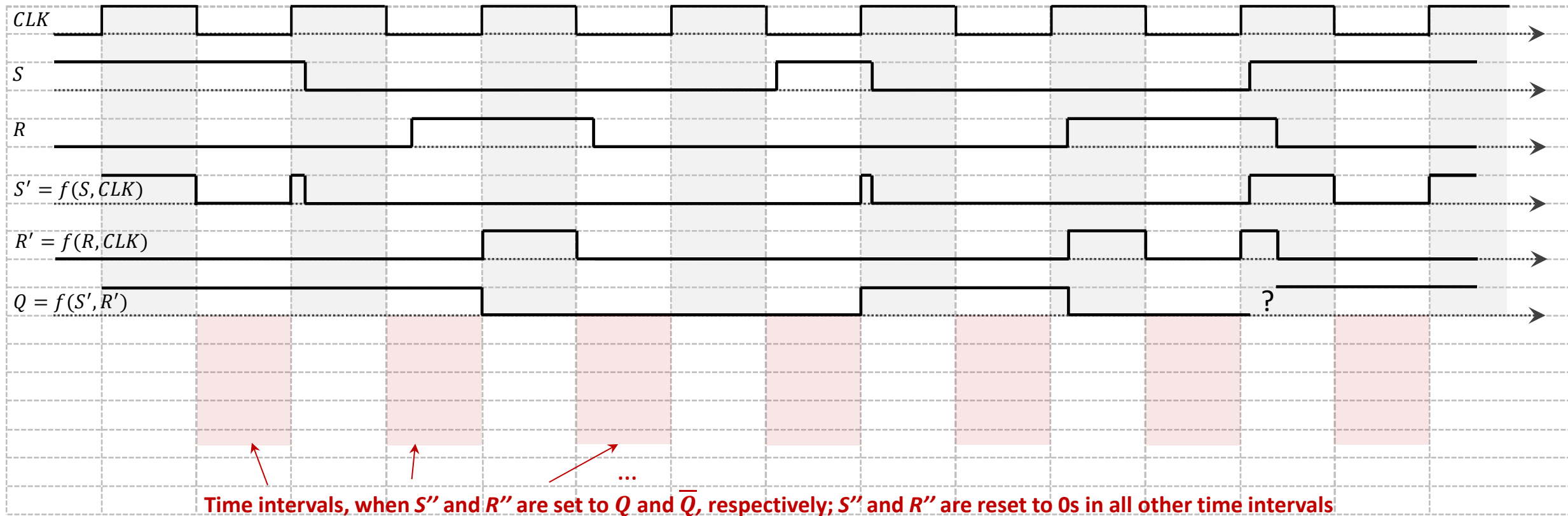
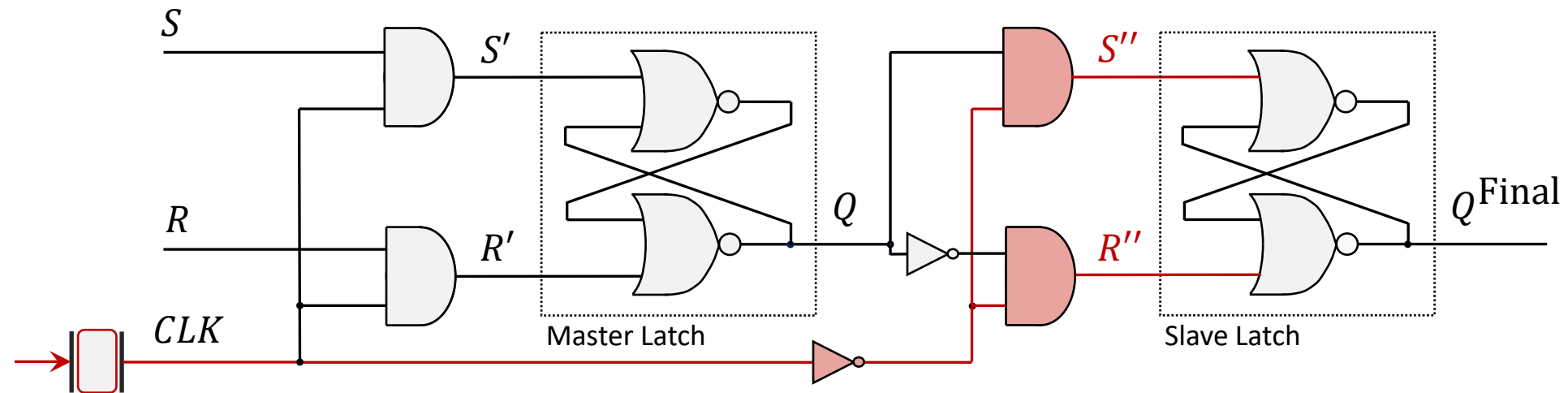
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



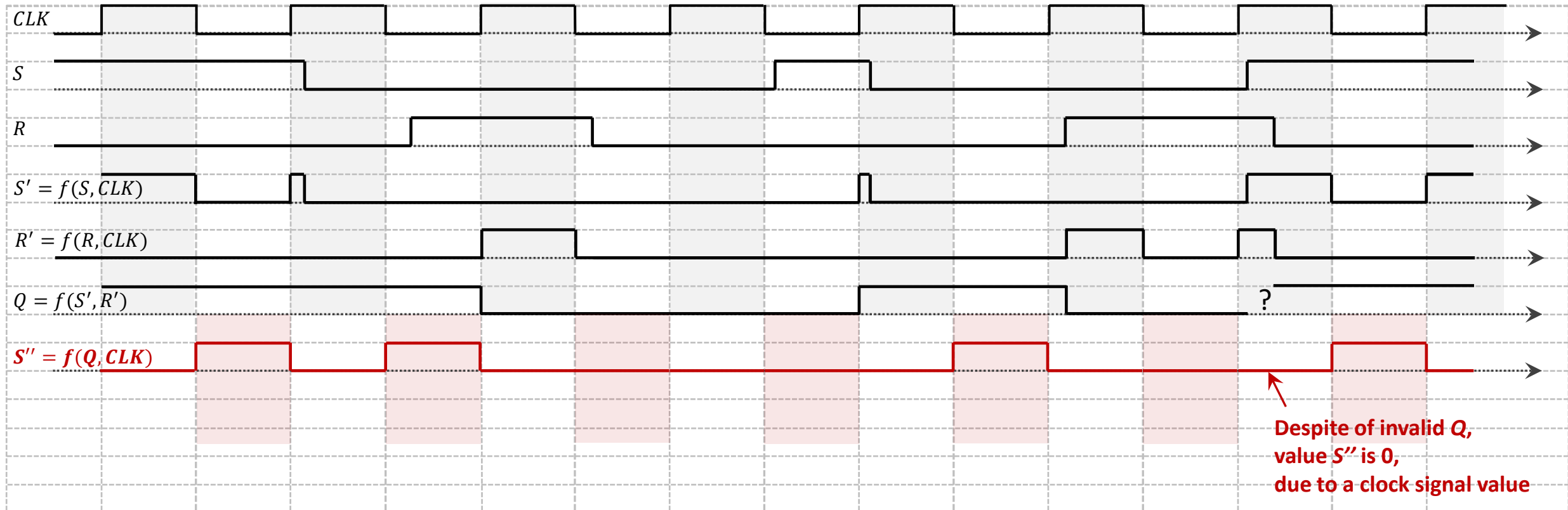
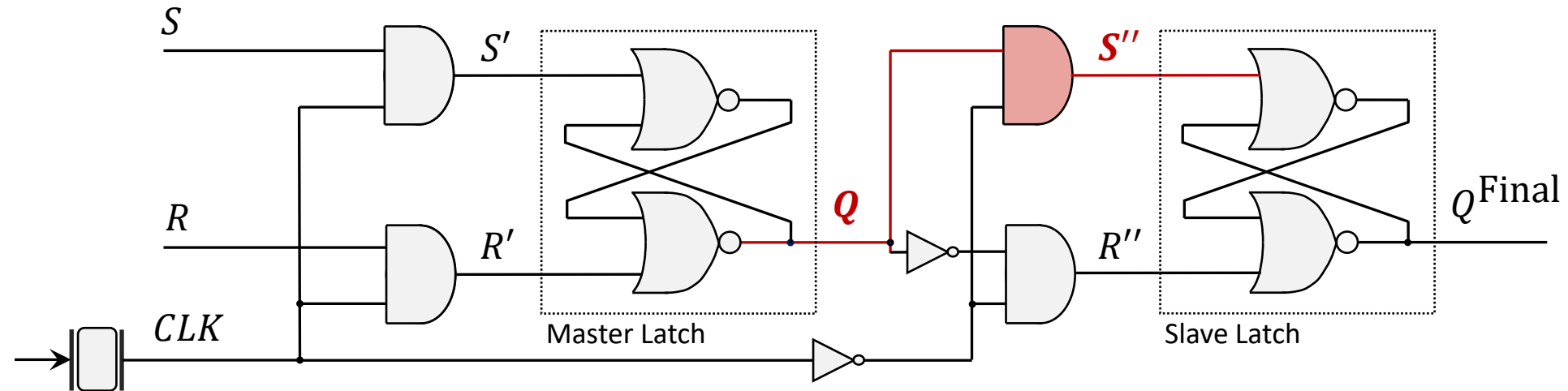
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



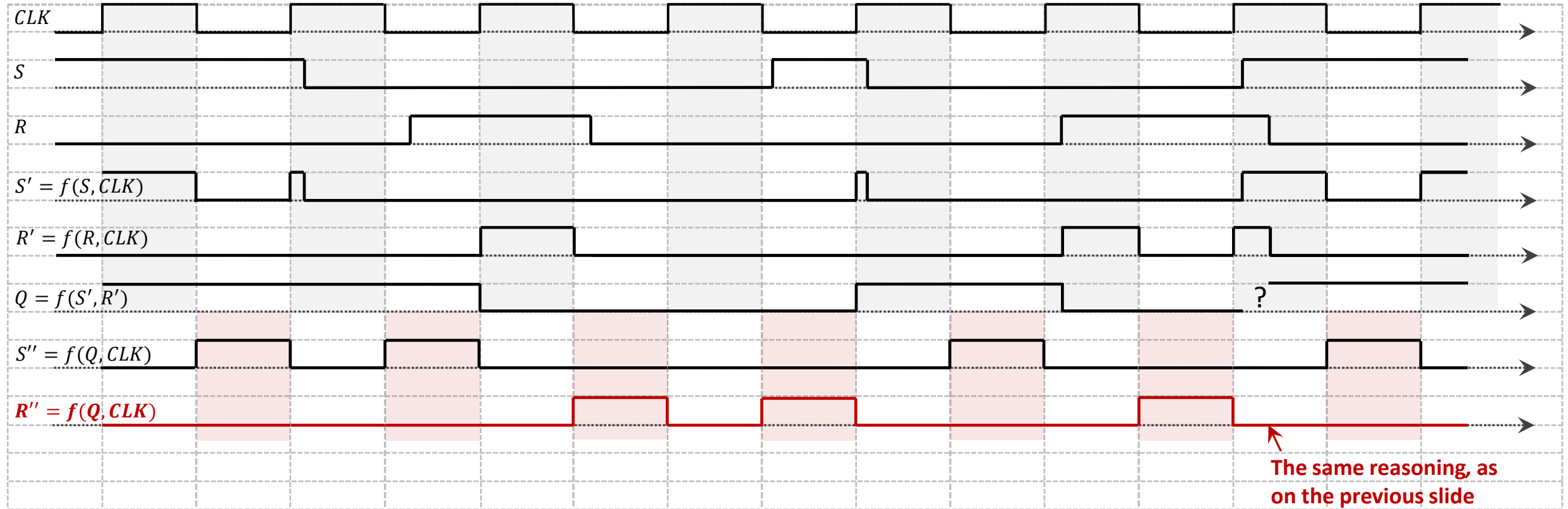
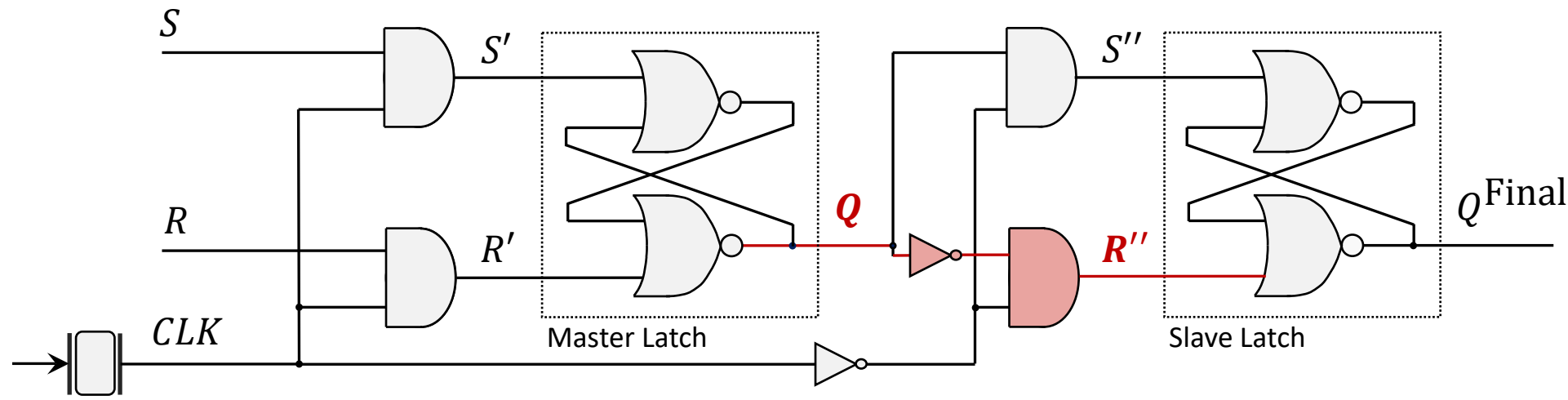
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



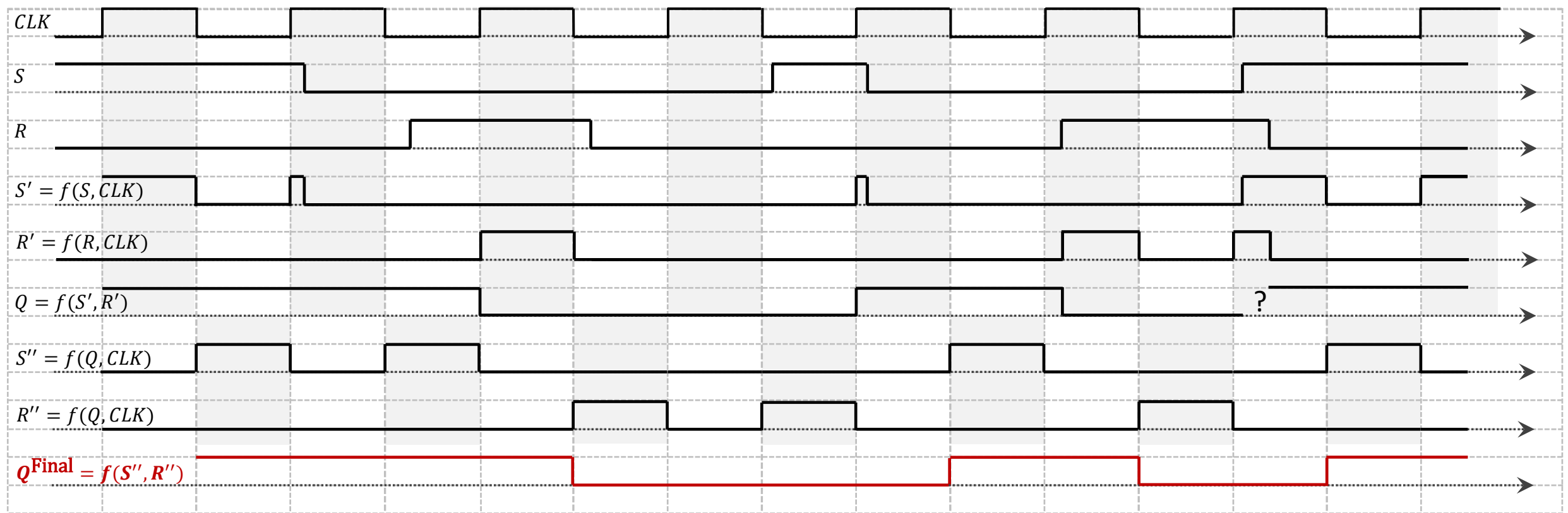
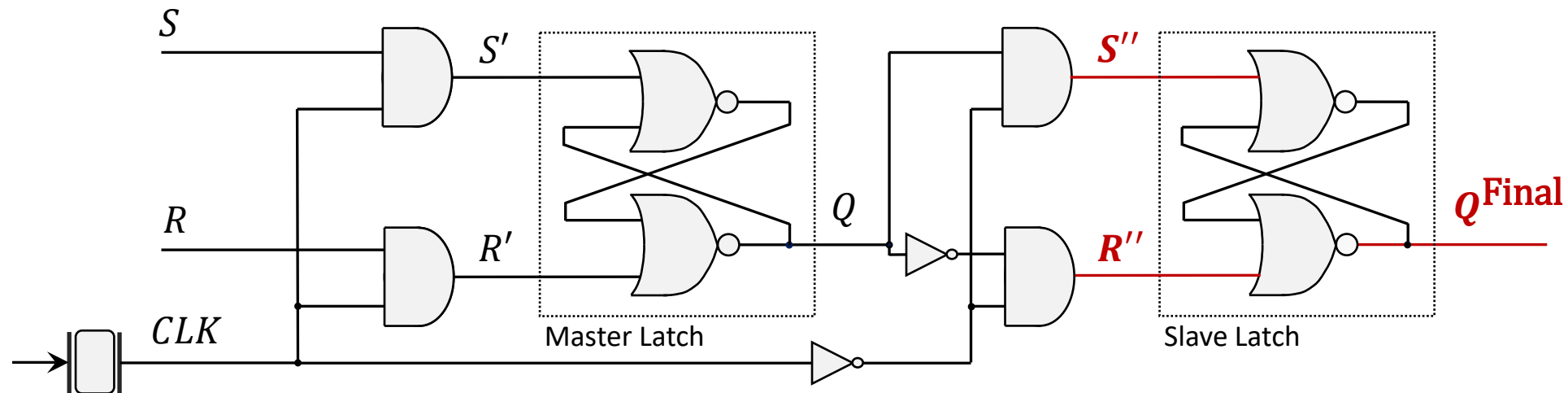
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



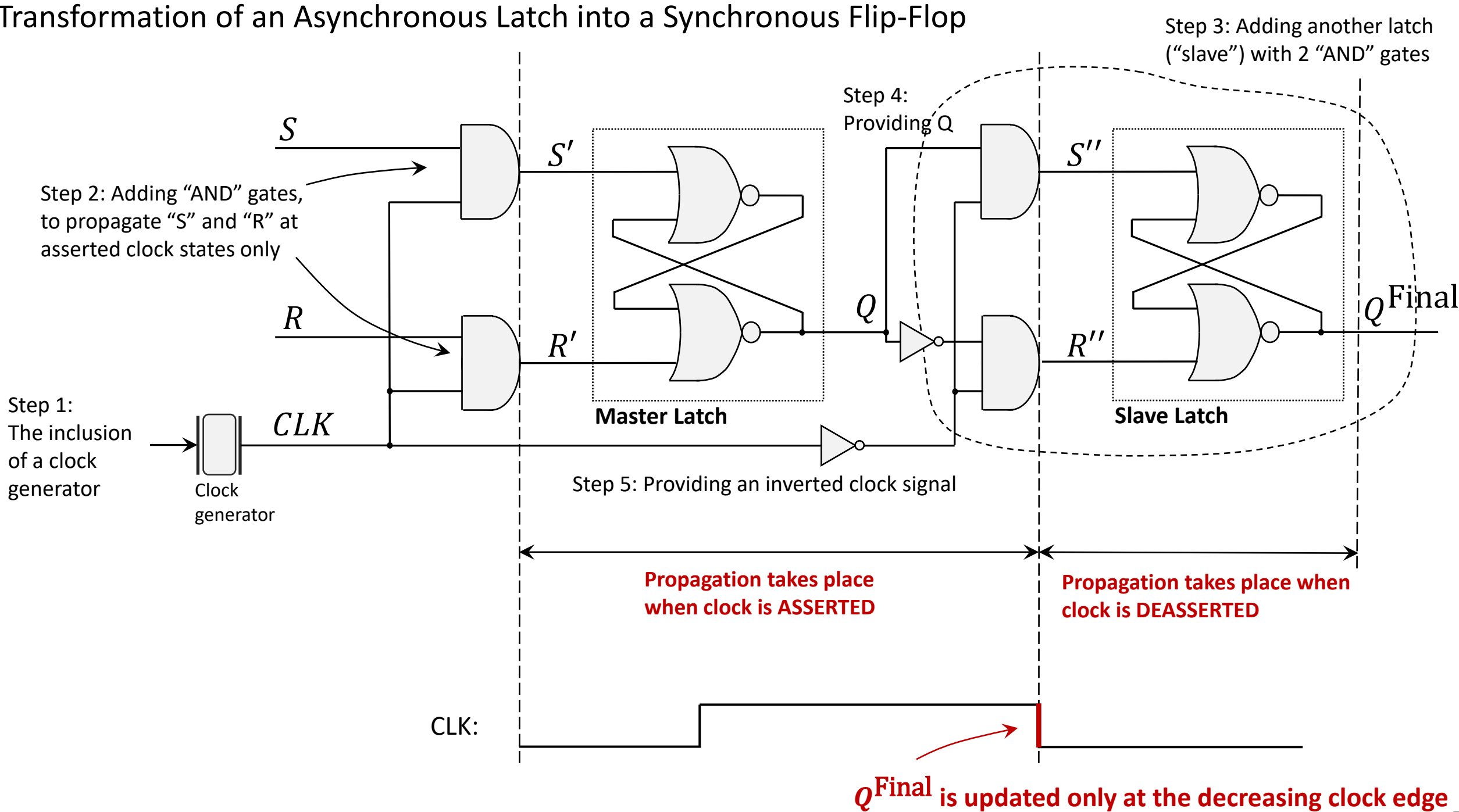
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop

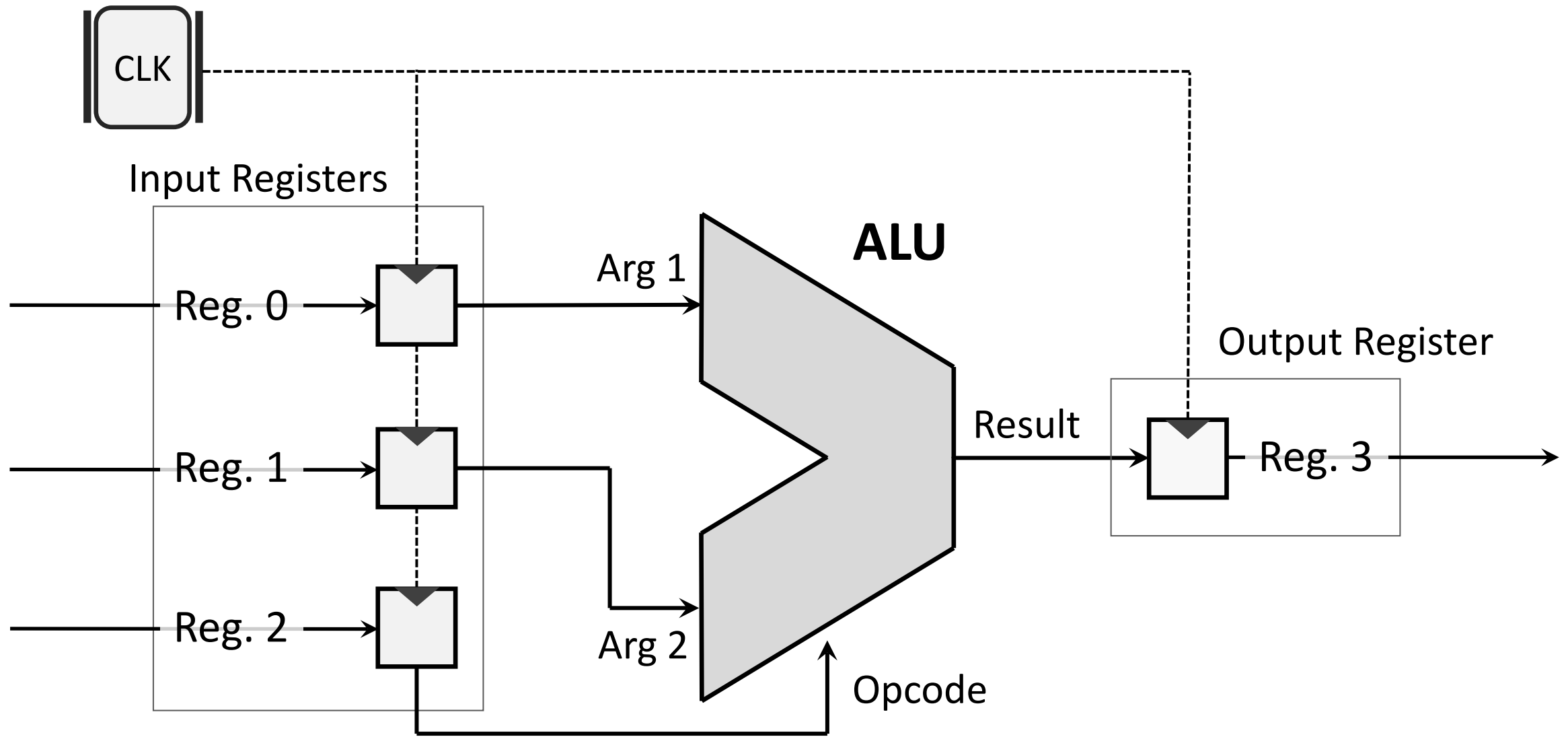


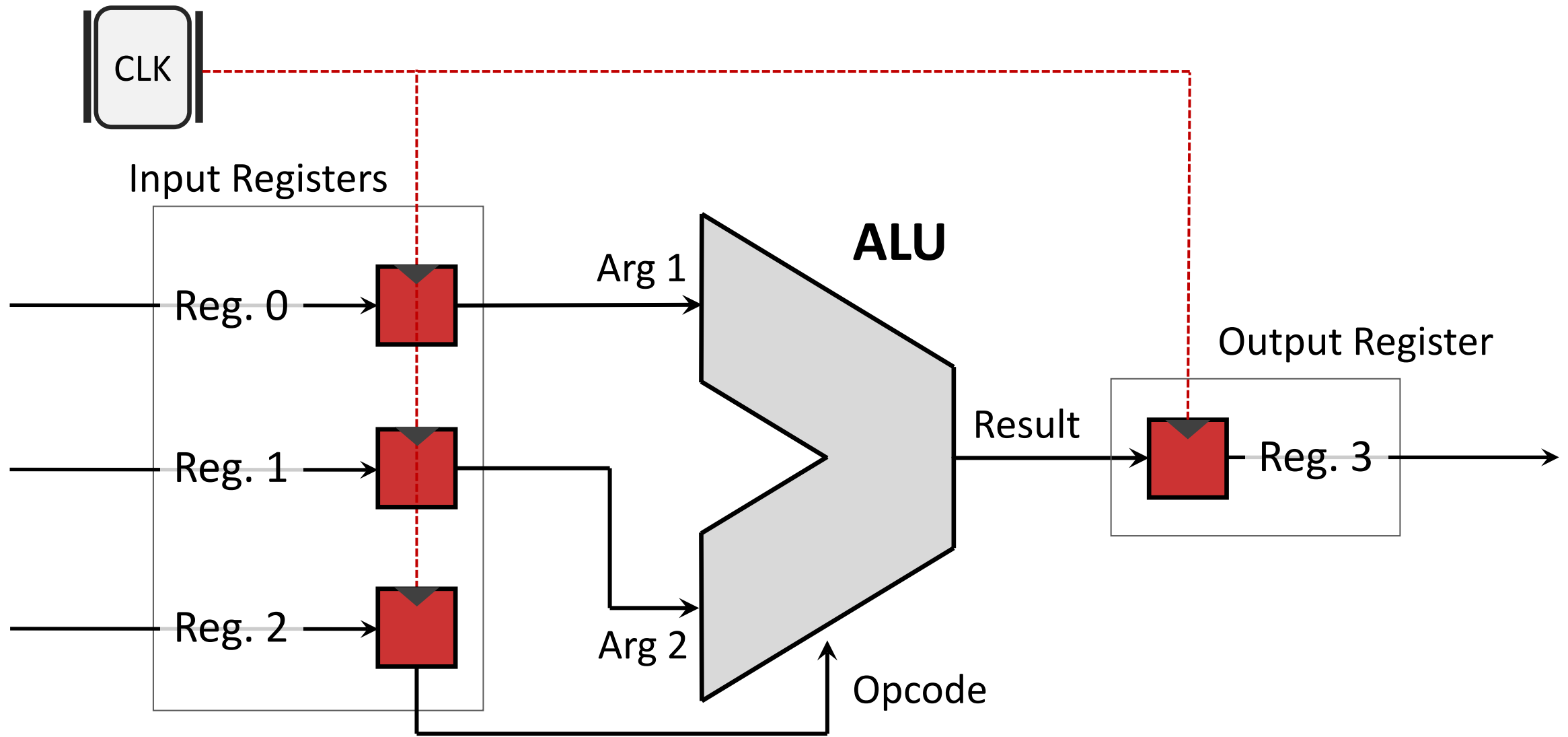
Transformation of an Asynchronous Latch into a Synchronous Flip-Flop



Transformation of an Asynchronous Latch into a Synchronous Flip-Flop







Registers are constructed of Flip-Flops

