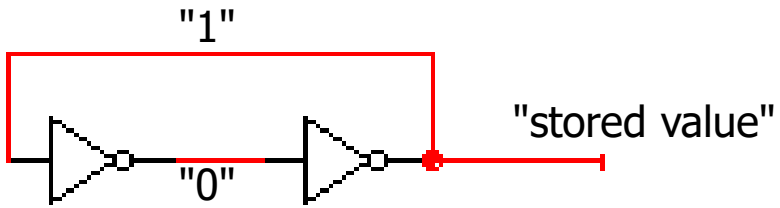
SEQUENTIAL LOGIC DESIGN

Introduction

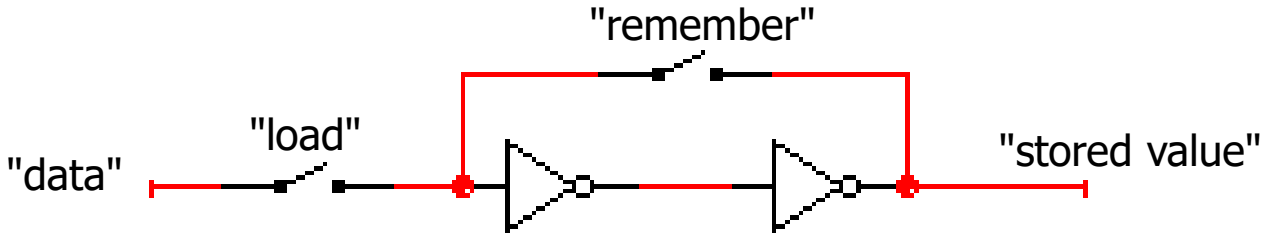
- Combinational Circuits
  - Without memory
  - Outputs depend only on current inputs
  - Output determined after sufficient time has elapsed
- Sequential Circuits
  - Have memory
  - Outputs depend on inputs and previous outputs
  - Previous output stored even after waiting

Simple Sequential Circuits

- Two inverters form a static memory cell
  - The previous output is stored by passing it back into the system
  - Will hold value as long as it has power applied



- How to get a new value into the memory cell?
  - Selectively break feedback path
  - Load new value into cell



Sequential Circuits

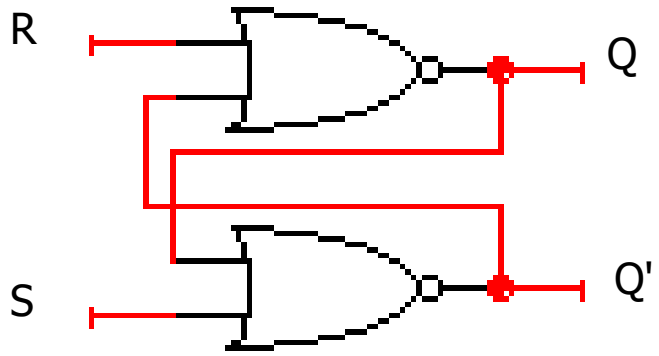
- Latches
- Flip-flops
- Registers

Latches

- A latch is a memory element in which the output is equal to the value of the stored state inside the element and the state is changed whenever the appropriate inputs change.

Set-Reset Latch

- Built from cross-coupled NOR gates
- Ability to force output to 0 (reset=1) or 1 (set=1)



State Machines

- A sequential circuit is described as a finite-state machine
  - A set of states
  - A function to determine the next state
  - A function to determine the output
    - Functions can still be represented by a truth table

Behavior of the S-R Latch

- States: {hold, set, reset, unstable}
- Next-State function:

S	R	State
0	0	hold
0	1	0
1	0	1
1	1	unstable

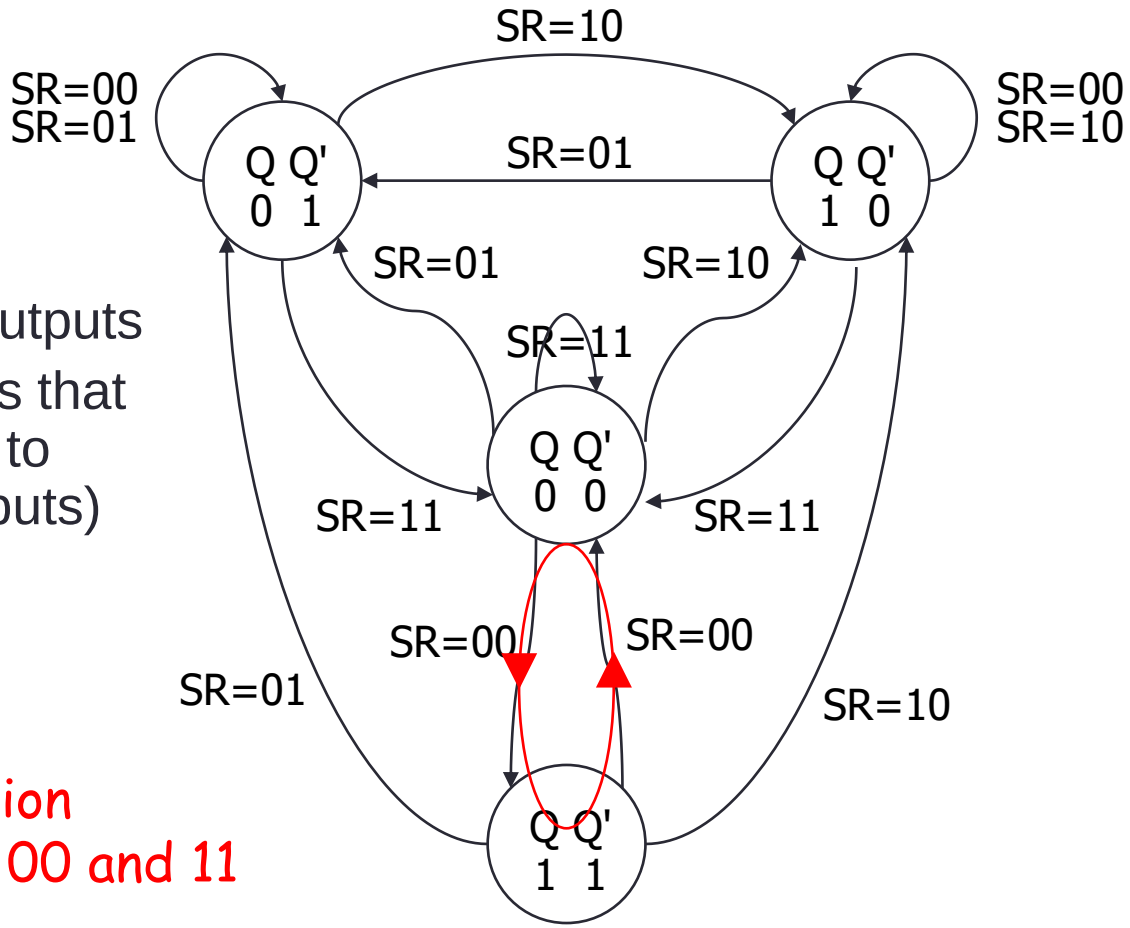
Behavior of the S-R Latch

- States: {hold, set, reset, unstable}
- Output function:

S	R	Qin	Qout
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	?
1	1	1	?

Behavior of the S-R Latch

- State Diagram

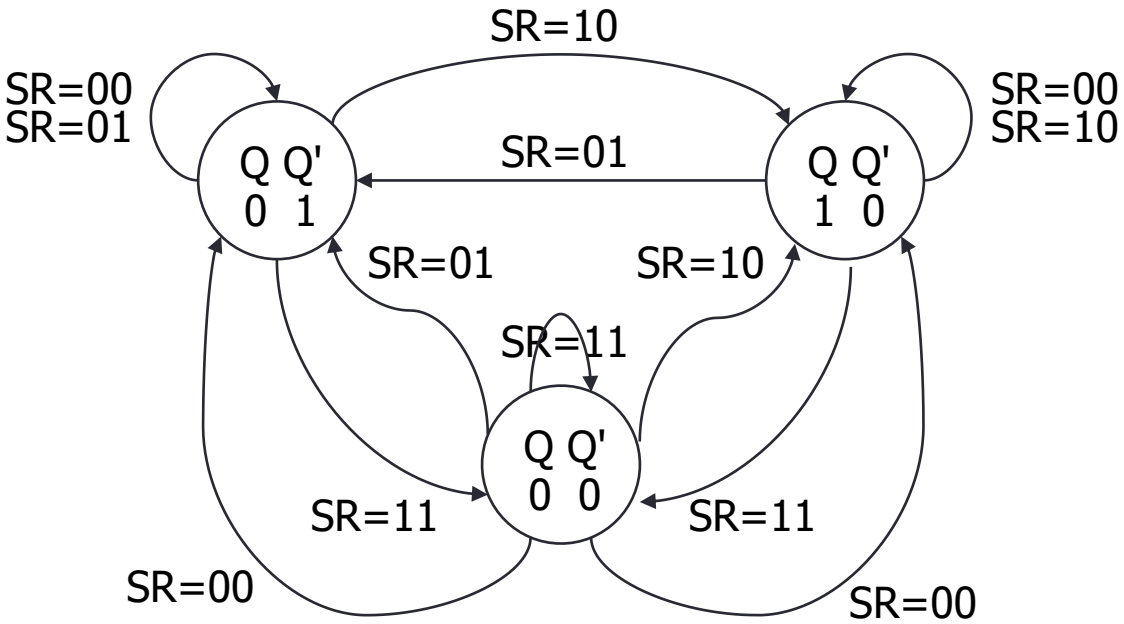


- States: possible outputs
- Transitions: events that cause the system to change states (inputs)

possible oscillation
between states 00 and 11

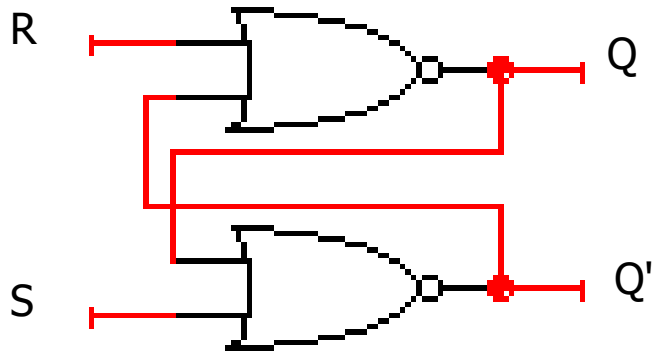
Behavior of the S-R Latch

- Very difficult to observe R-S latch in the unstable state
  - One of R or S usually changes first
- Ambiguously returns to state 0-1 or 1-0
  - A so-called "race condition"
  - Or non-deterministic transition



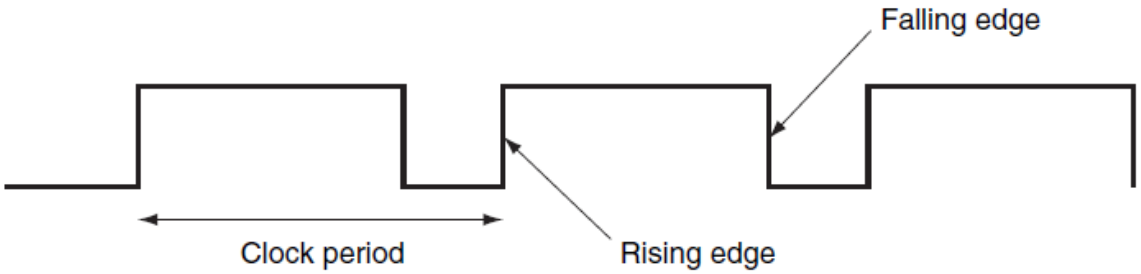
Unclocked S-R Latch

- In an unclocked circuit the outputs change whenever the inputs change.



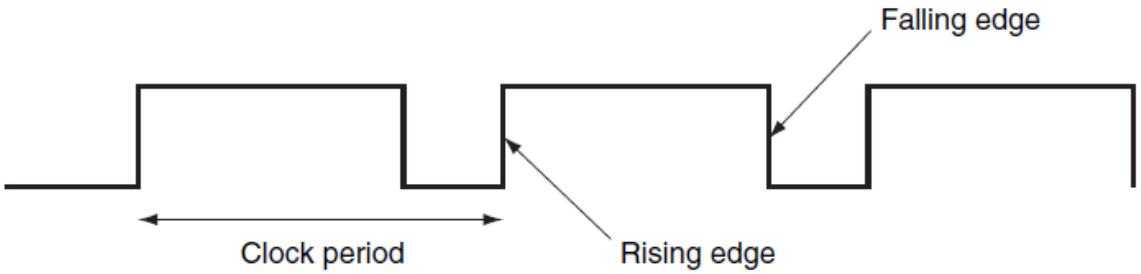
Clocks

- Clocks are regular periodic signals
  - Clock period is divided into two portions
    - When the clock is “high”, also called the duty cycle
    - When the clock is “low”
  - State changes occur on a clock edge
    - When the clock moves from low to high or from high to low



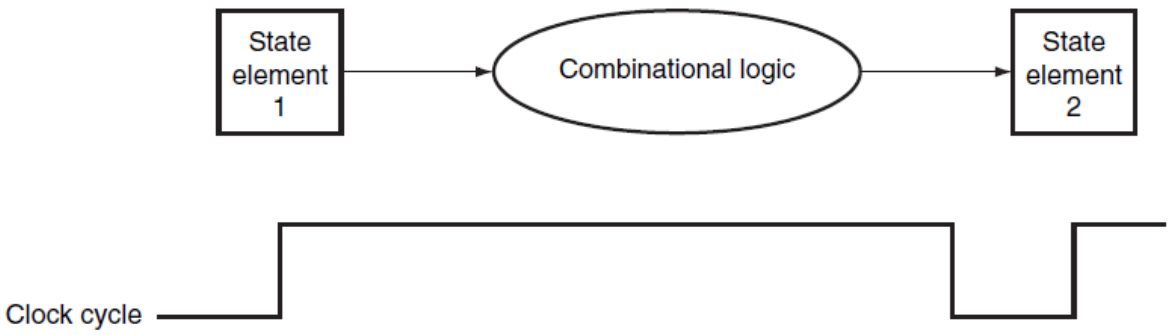
Edge-Triggered Methodology

- Either the rising edge or the falling edge is active
  - Determined by implementation
  - Does not affect design
- Elements change on the active clock edge
  - All inputs sampled at the same time
  - All inputs must be valid (steady) at the active clock edge
- Clocked systems are also called synchronous systems.



Sequential Logic Design

- Combinational logic elements are still used
- State elements provide valid inputs to the combinational logic block.
- Clock period must be long enough to allow all the signals in the combinational logic block to settle

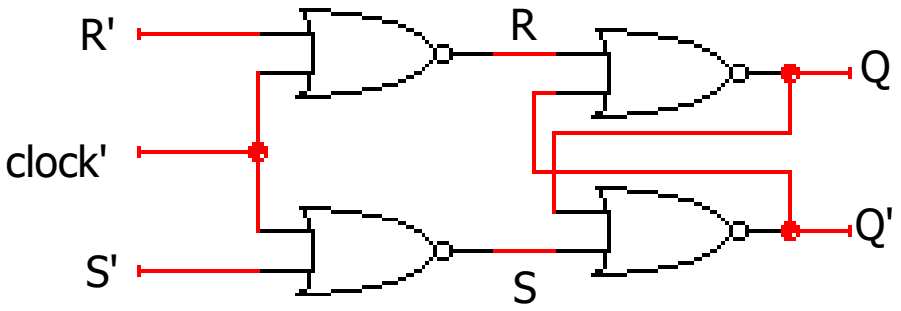


Steady-State Abstraction

- All real circuits need time to update
  - The outputs do not change instantaneously after an input change
- A fundamental abstraction of digital design is to reason about steady-state behaviors
  - Look at outputs only after sufficient time has elapsed for the system to make its required changes and settle down
  - Memory of a system is represented as its state
  - Changes in system state are only allowed to occur at specific times

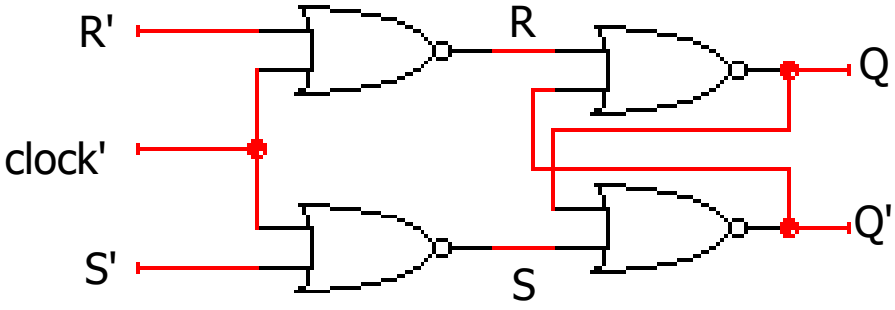
Clocked S-R Latch

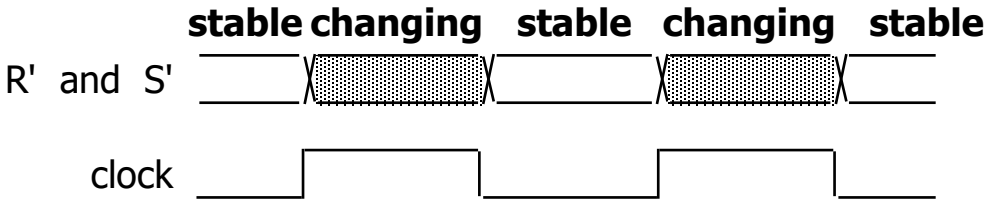
- In a clocked circuit, we control when the inputs are sampled.



Clocked S-R Latch

- Controlling an R-S latch with a clock
  - Can't let R and S change while clock is active
  - Only have half of clock period for signal changes to propagate
  - Signals must be stable for the other half of clock period



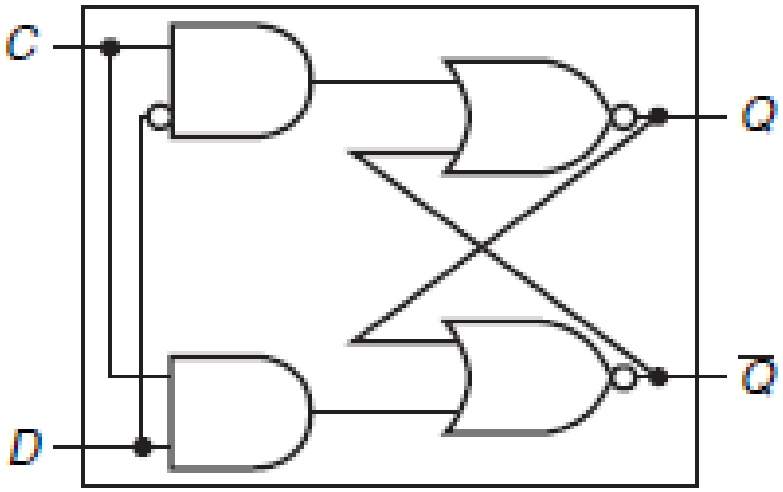


Flip-Flops

- A flip-flop is a memory element for which the output is equal to the value of the stored state inside the element and for which the internal state is changed only on a clock edge.
  - Often built from clocked latches

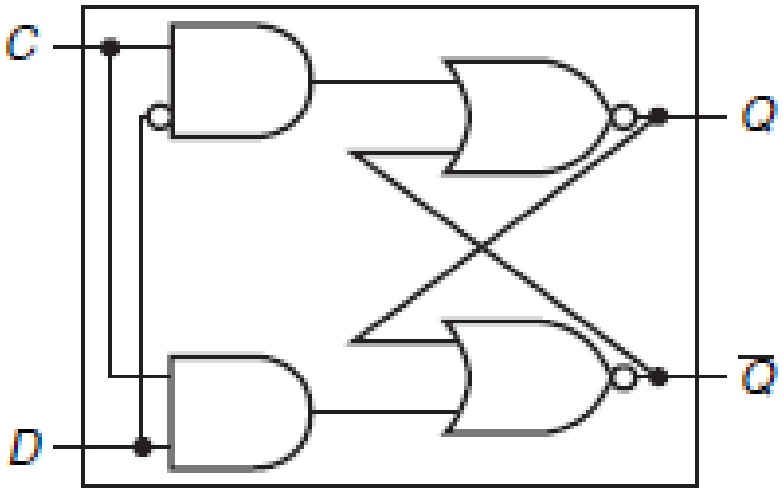
D Flip-Flop

- 2 Inputs
  - D, the data to be stored
  - C, the clock signal
- 2 Outputs
  - Q, the internal state
  - Q', the complement of Q



D Flip-Flop

- When the clock is asserted, the flip-flop is “open”
  - If D is 1, the S-R latch is “set”
  - If D is 0, the S-R latch is “reset”
- When the clock is deasserted, the flip-flop is “closed”
  - The S-R latch is “holding”

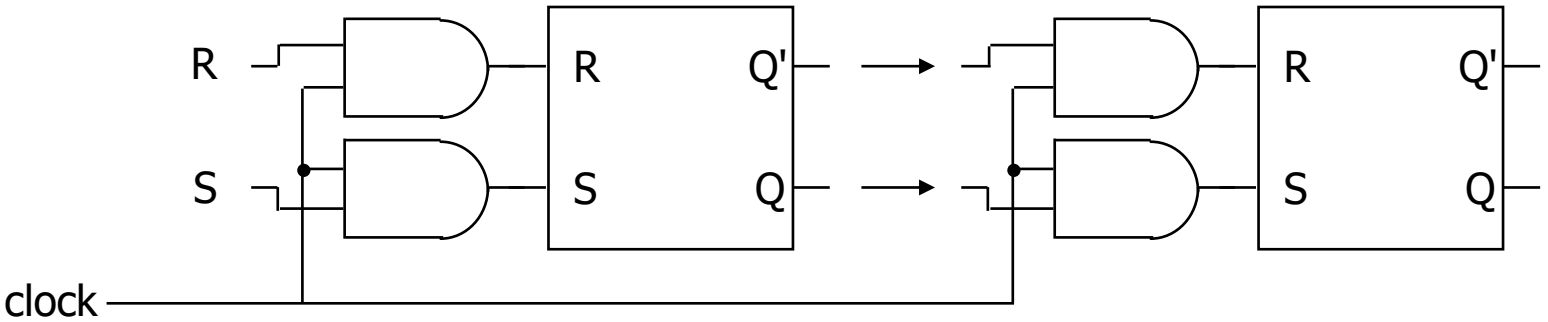


D Flip-Flop

- Flip-Flops can be built to trigger on either the rising or falling clock edge
  - Remember this does not affect the design
- Flip-Flops can be build to trigger on the offset edge
  - If all Flip-Flops trigger on the positive edge, we can build one to trigger on the negative edge
  - We use cascading logic blocks

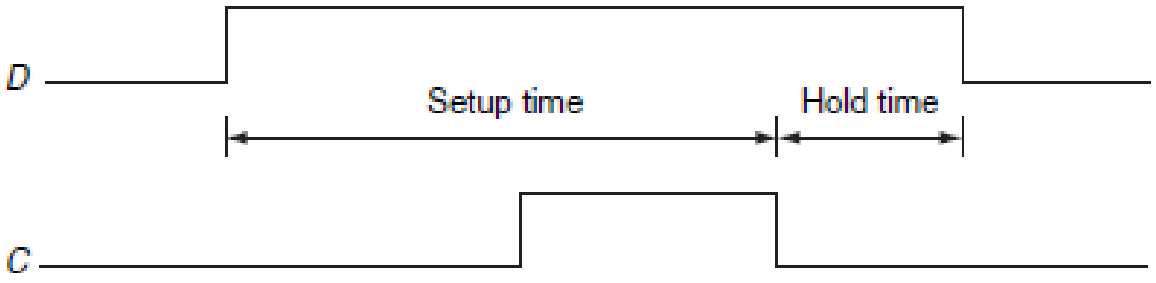
Cascading Logic Circuits

- Connect output of one latch to input of another
- How to stop changes from racing through chain?
  - Need to control flow of data from one latch to the next
  - Advance from one latch per clock period
  - Must respect setup and hold time constraints to successfully capture input



Timing Methodologies

- The set up time is the minimum time that the input must be valid before the clock edge.
- The hold time is the minimum time during which it must be valid after the clock edge.

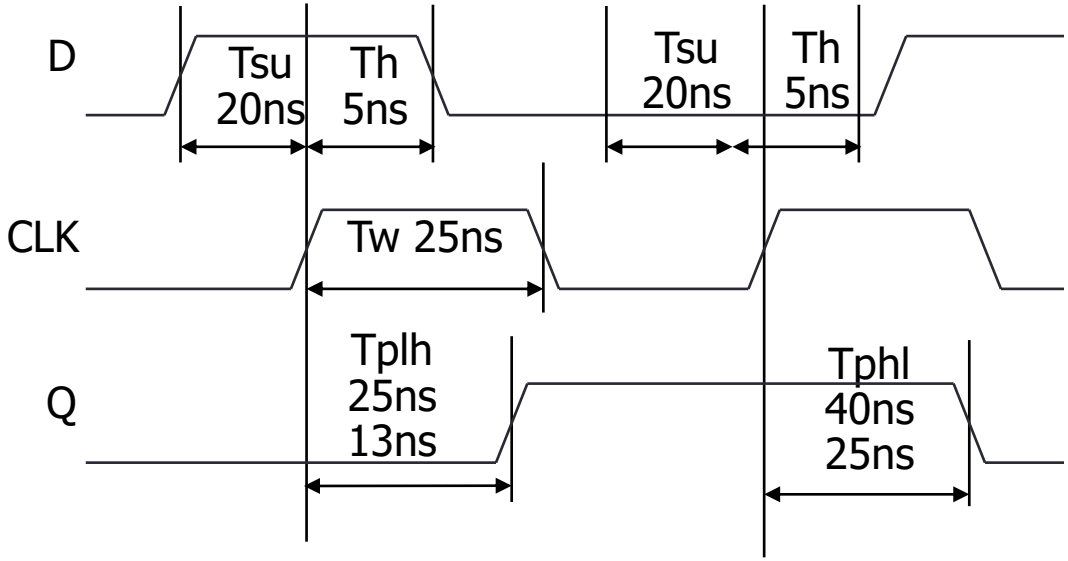


Timing Methodologies

- Rules for interconnecting components and clocks
  - Guarantee proper operation of system when strictly followed
- Approach depends on building blocks used for memory elements
  - Focus on systems with edge-triggered flip-flops
- Basic rules for correct timing:
  - (1) Correct inputs, with respect to time, are provided to the flip-flops
  - (2) No flip-flop changes state more than once per clocking event

Typical Timing Specifications

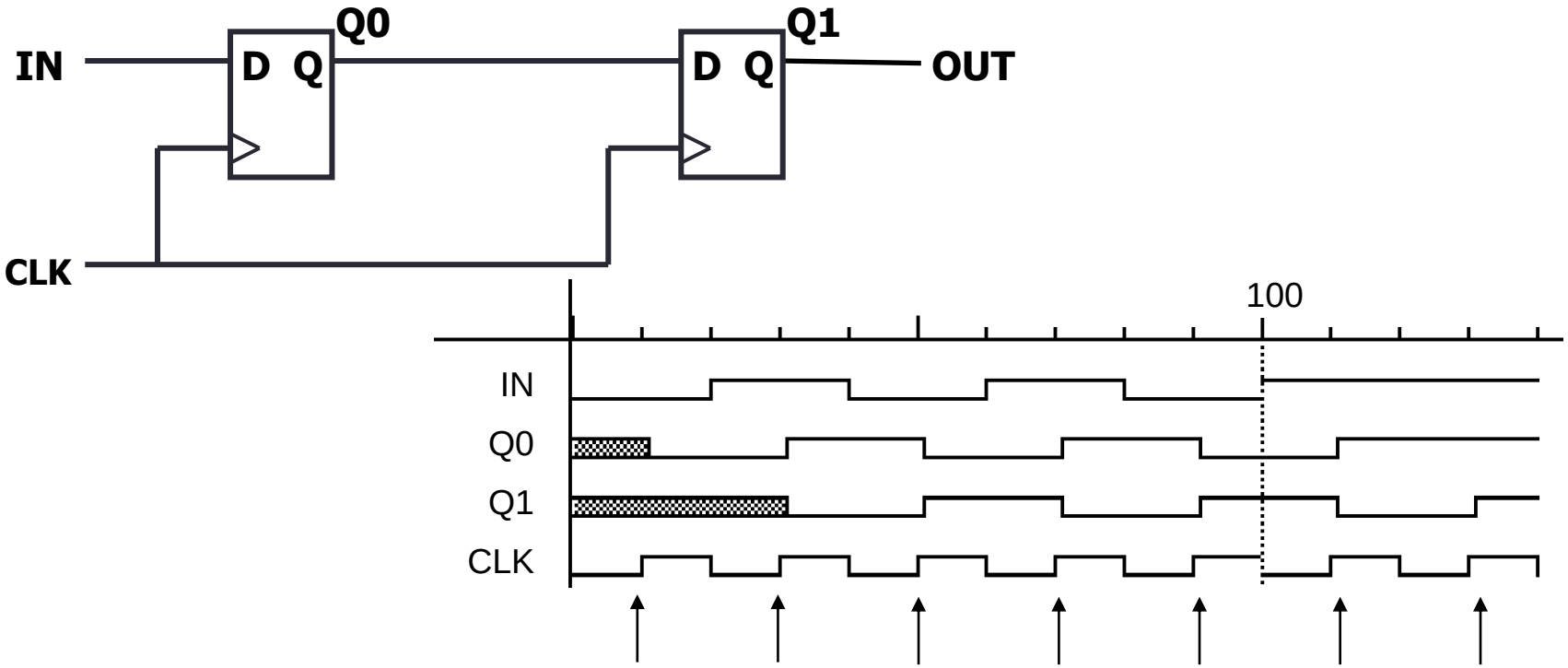
- Positive edge-triggered D flip-flop
  - Setup and hold times
  - Minimum clock width
  - Propagation delays (low to high, high to low, max and typical)



all measurements are made from the clocking event:
the rising edge of the clock

Cascading Edge-triggered Flip-Flops

- Consider setup/hold/propagation delays
  - prop must be > hold

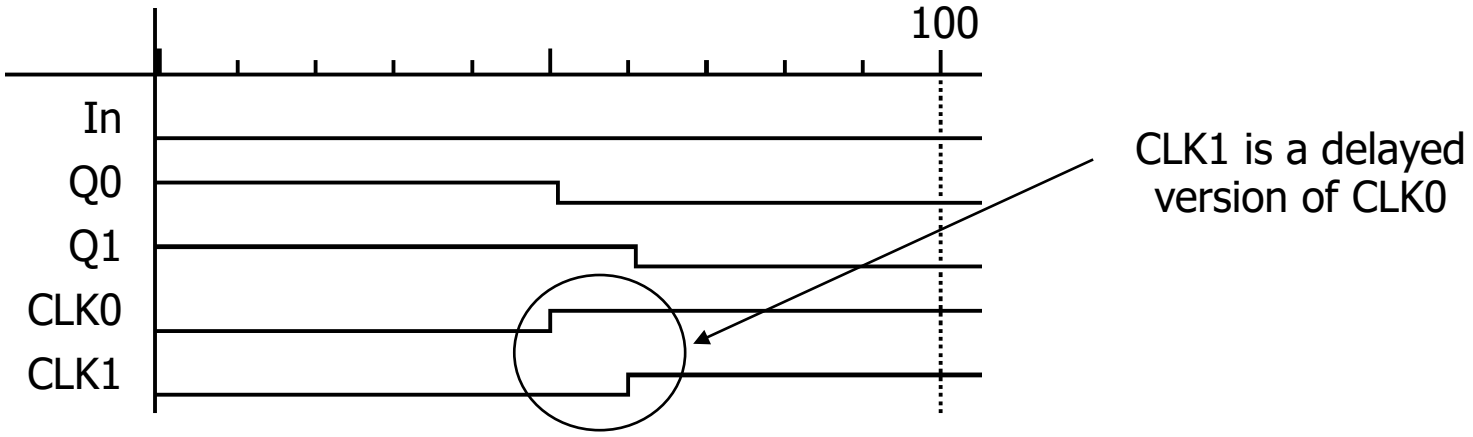


Clock Skew

- Correct behavior assumes next state of all storage elements determined by all storage elements at the same time
- This is difficult in high-performance systems because time for clock to arrive at flip-flop is comparable to delays through logic
- Clock skew is the difference in time between when two state elements see a clock edge

Clock Skew

- If the clock skew is large enough, it may be possible for a state element to change and cause the input to another flip-flop to change before the clock edge is seen by the second flip-flop.
  - To avoid incorrect operation, we must increase the clock period to allow for the maximum clock skew.



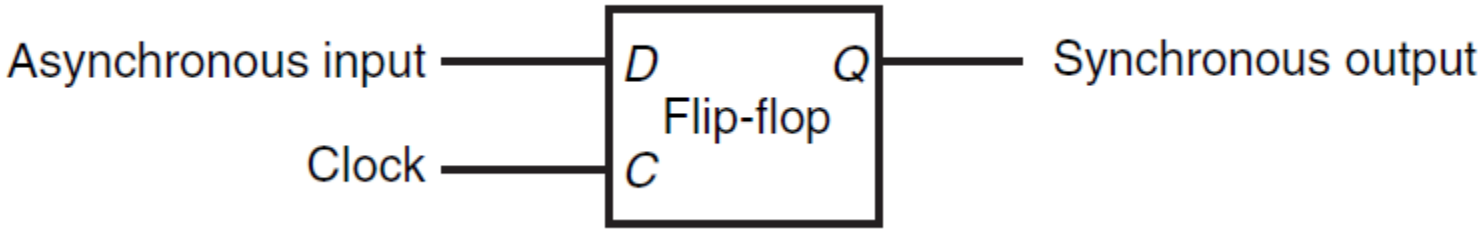
original state: IN = 0, Q0 = 1, Q1 = 1
due to skew, next state becomes: Q0 = 0, Q1 = 0, and not Q0 = 0, Q1 = 1

Asynchronous Inputs

- Clocked synchronous circuits
  - Inputs, state, and outputs sampled or changed in relation to a common reference signal (called the clock)
- Asynchronous circuits
  - Inputs, state, and outputs sampled or changed independently of a common reference signal
  - Some system inputs must be asynchronous
    - reset signal, memory wait, user input
  - Unstable states are a major concern
- Asynchronous inputs to synchronous circuits
  - Because synchronous inputs are greatly preferred, we use a synchronizer

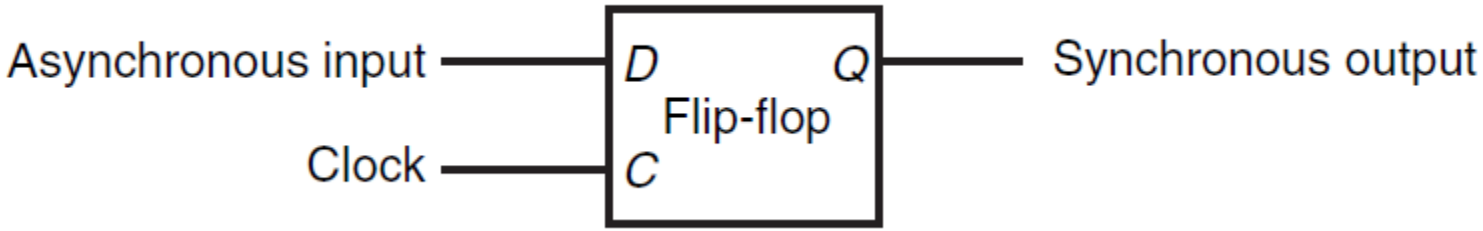
Synchronizer

- Use a synchronizer to translate asynchronous input



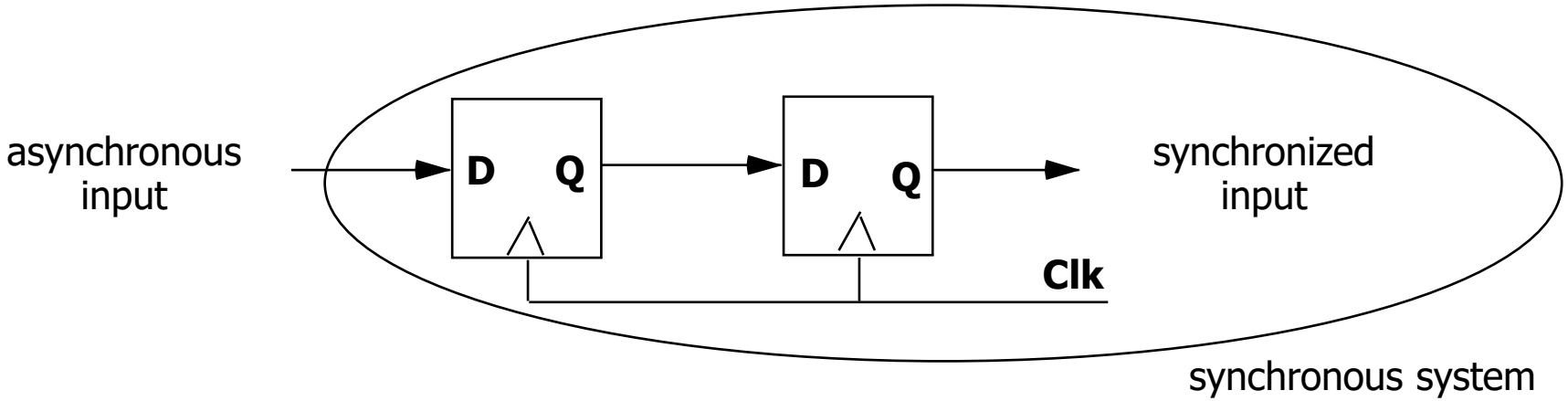
Synchronizer

- What happens if the asynchronous input is transitioning between 0 and 1?
  - The signal is not stable for the required setup and hold times
  - The synchronizer may enter a metastable state
  - The output will be neither 0 or 1
  - This situation is called synchronizer failure



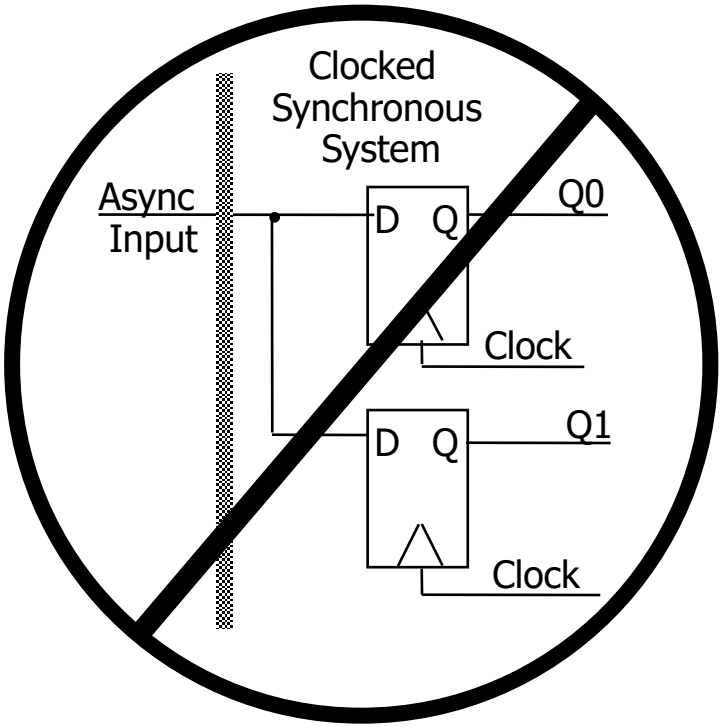
Synchronization Failure

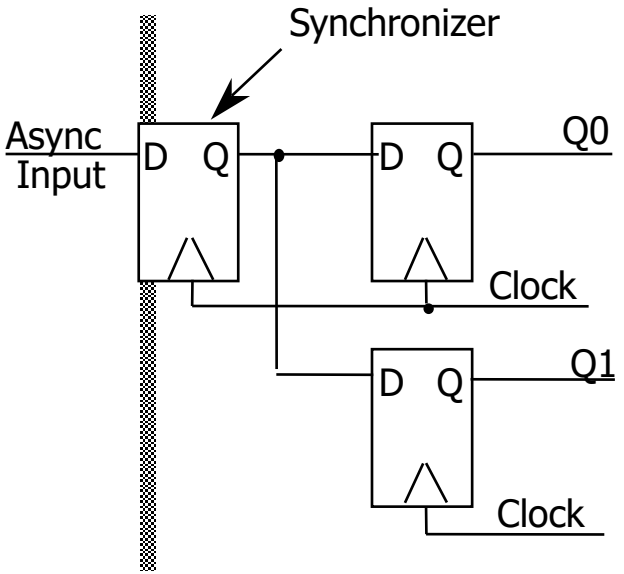
- Probability of failure can never be reduced to 0, but it can be reduced:
  - slow down the system clock: this gives the synchronizer more time to decay into a steady state; synchronizer failure becomes a big problem for very high speed systems
  - Narrow the “unknown” range of values: this makes for a very sharp "peak" upon which to balance
  - cascade two synchronizers: this effectively synchronizes twice (in subsequent clock cycles)



Handling Asynchronous Inputs

- Never allow asynchronous inputs to fan-out to more than one flip-flop
  - Synchronize as soon as possible and then treat as synchronous signal



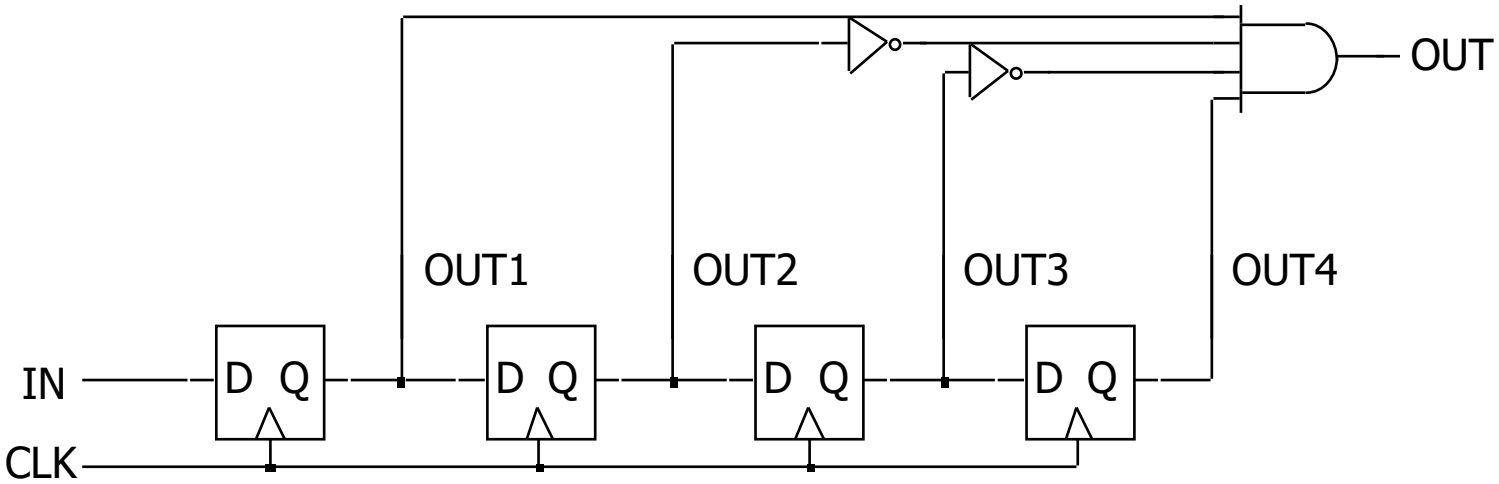


Sequential Hardware Elements

- Collections of flip-flops with similar controls and logic
  - Values in individual flip-flops are related
  - Shared clock signal
  - Similar logic at each stage
- Examples
  - Pattern Recognizer
  - Counter
  - Shift register
  - Data Register
  - Register File

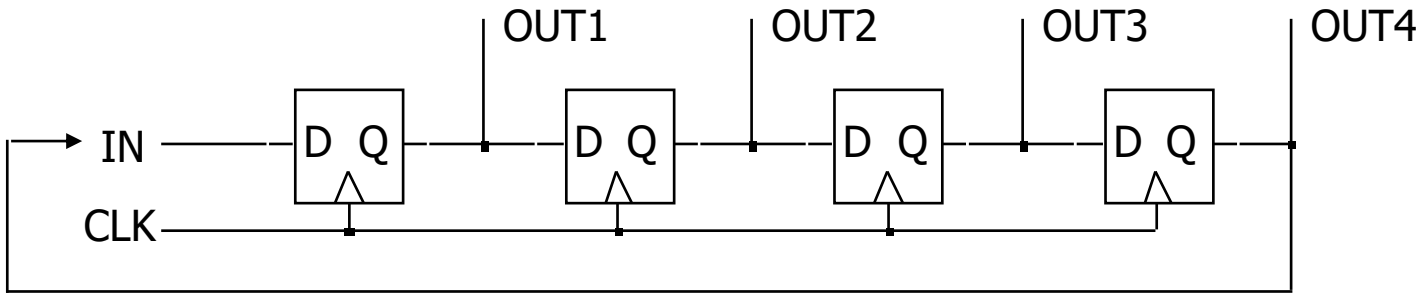
Pattern Recognizer

- Combinational function of input samples
  - In this case, recognizing the pattern 1001 on the single input signal

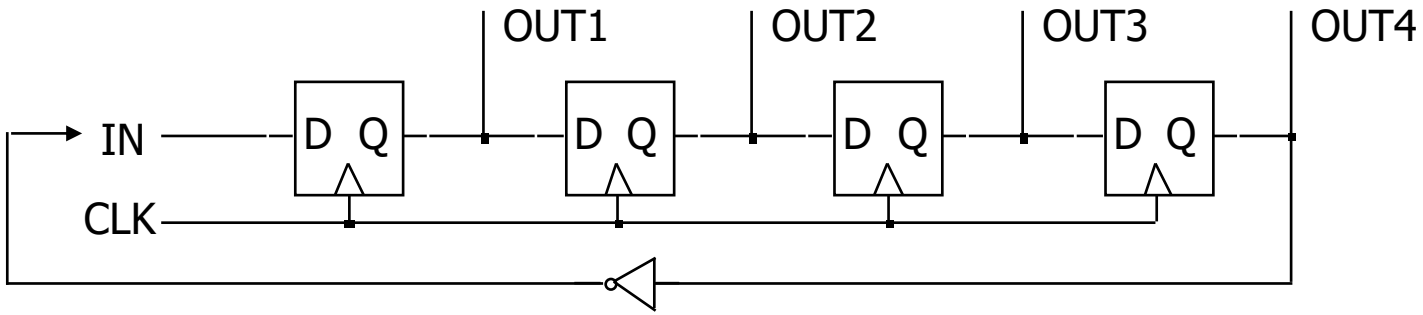


Counters

- Sequences through a fixed set of patterns
  - In this case, 1000, 0100, 0010, 0001
  - If one of the patterns is its initial state (by loading or set/reset)

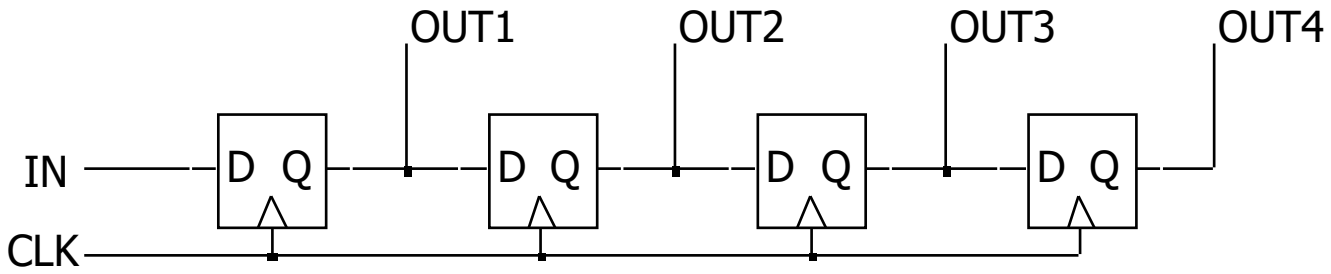


- Mobius (or Johnson) counter
  - In this case, 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000



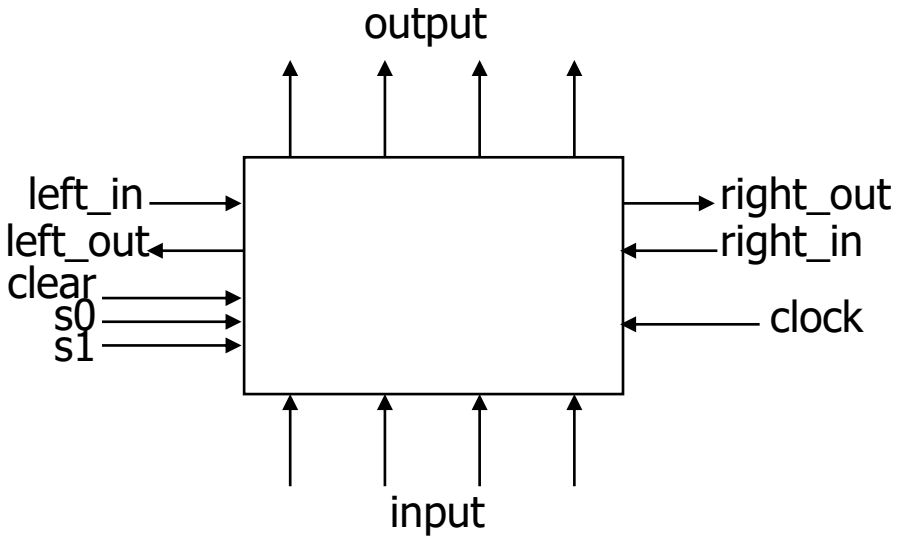
Shift Register

- Holds samples of input
  - Store last 4 input values in sequence
  - 4-bit shift register:



Universal Shift Register

- Holds 4 values
  - Serial or parallel inputs
  - Serial or parallel outputs
  - Permits shift left or right
  - Shift in new values from left or right



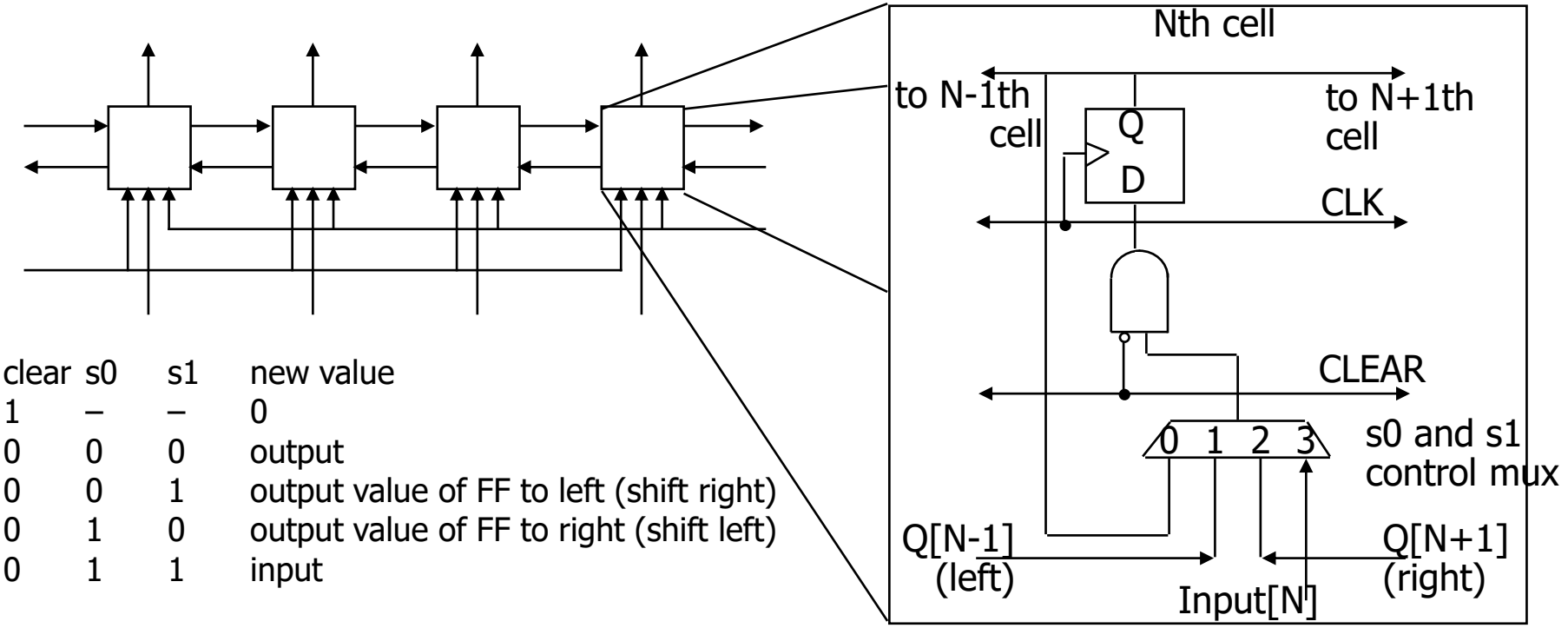
clear sets the register contents and output to 0

s1 and s0 determine the shift function

s0	s1	function
0	0	hold state
0	1	shift right
1	0	shift left
1	1	load new input

Design of Universal Shift Register

- Consider one of the four flip-flops
  - New value at next clock cycle:

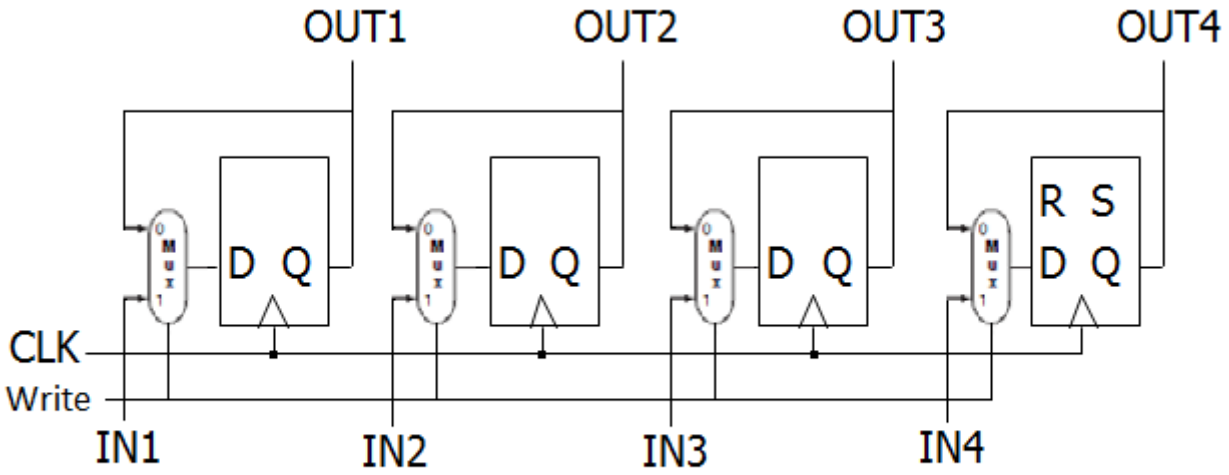


clear	s0	s1	new value
1	–	–	0
0	0	0	output
0	0	1	output value of FF to left (shift right)
0	1	0	output value of FF to right (shift left)
0	1	1	input

Data Register

- Capable of holding a single binary value
- Loads a new value when write is enabled

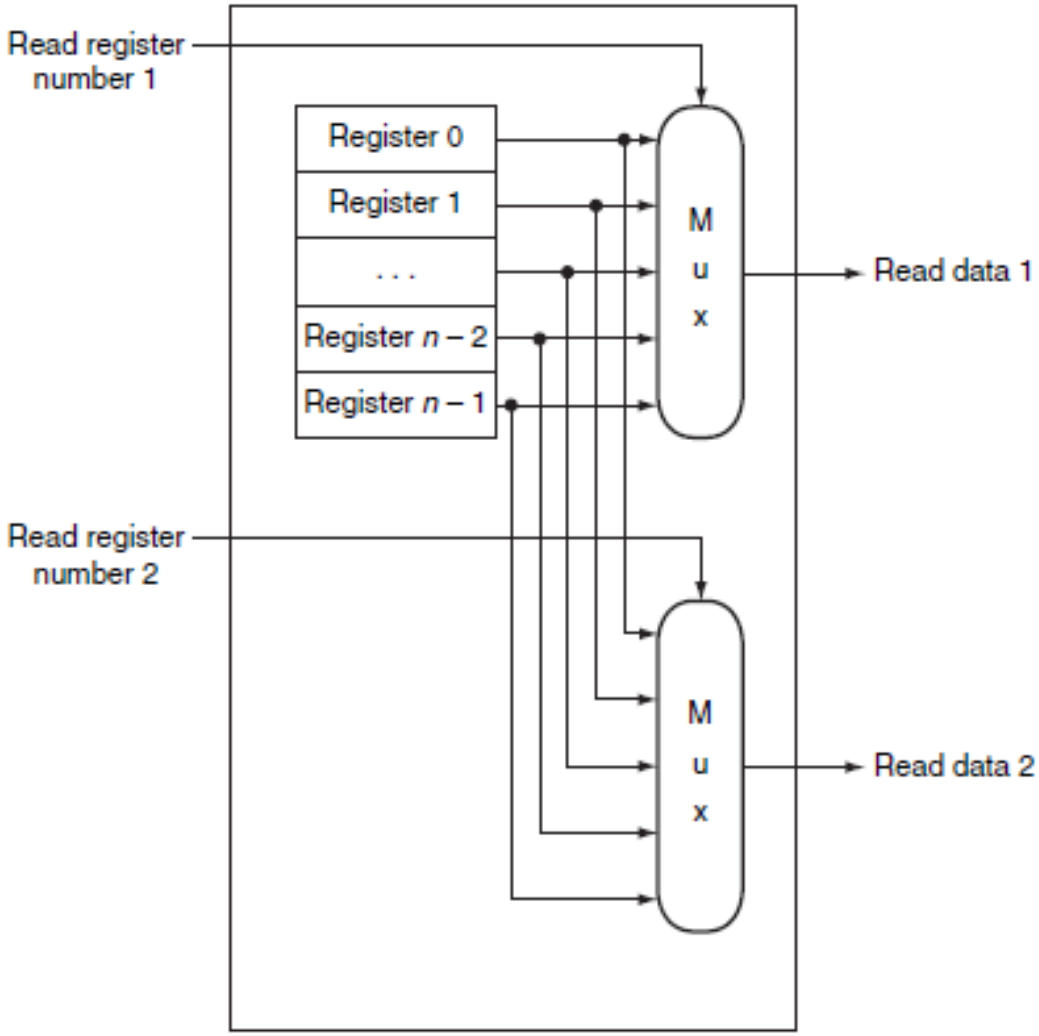
- 4-bit Data Register



Register Files

- A register file consists of a set of registers that can be read and written by supplying a register number to be accessed.

Register Files: Read



Register Files: Write