

UNIT 4

Counters & Registers

(16 Hours) — Digital Logic

This unit covers the building blocks of sequential digital circuits — flip-flops and their triggering behaviour, the design and analysis of asynchronous and synchronous counters, common counter applications, and the four fundamental modes of shift registers.

4.1 RS, JK, JK Master–Slave, D & T Flip-Flops

A flip-flop is a bistable multivibrator — a sequential logic element that stores one bit of information and changes state only in response to a clock signal. Flip-flops are the fundamental memory elements from which counters and registers are built.

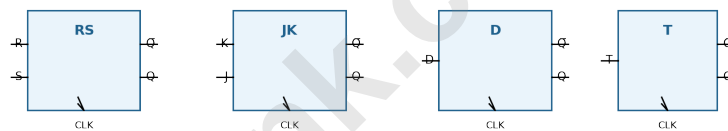


Fig. 4.1 — Standard block symbols for RS, JK, D and T flip-flops

RS (Set–Reset) Flip-Flop

The RS flip-flop has two inputs, S (Set) and R (Reset). When $S = 1$ and $R = 0$ the output Q is set to 1; when $S = 0$ and $R = 1$, Q is reset to 0. When both inputs are 0, the flip-flop holds its previous state. The condition $S = R = 1$ is not allowed in the basic NOR/NAND latch, since it produces an indeterminate (race) output.

S	R	Q(next)	Remark
0	0	Q (no change)	Hold
0	1	0	Reset
1	0	1	Set
1	1	Undefined / Invalid	Not allowed

JK Flip-Flop

The JK flip-flop removes the invalid condition of the RS flip-flop by feeding the outputs back to the input AND gates. When $J = K = 1$, the output toggles on every active clock edge instead of remaining undefined — this makes the JK flip-flop the most versatile of the basic flip-flop types.

J	K	Q(next)	Remark
0	0	Q (no change)	Hold
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}$ (complement)	Toggle

JK Master–Slave Flip-Flop

A single-edge JK flip-flop can suffer from race-around condition: if $J = K = 1$ and the clock pulse remains HIGH longer than the propagation delay of the gates, the output may toggle multiple times within a single pulse. The Master–Slave configuration solves this by cascading two latches — a Master latch that is enabled while CLK is HIGH, and a Slave latch enabled while CLK is LOW. Because the two latches are never transparent simultaneously, the output changes only once per clock cycle, on the falling edge.

Example 4.1 — Race-around condition

A single JK flip-flop has $J = K = 1$ and a clock pulse width of 50 ns. If each internal gate has a propagation delay of 10 ns, the flip-flop may toggle up to 5 times during one clock pulse, giving an unpredictable final state — this is the race-around problem that the Master–Slave design eliminates.

D (Data / Delay) Flip-Flop

The D flip-flop has a single data input D. On the active clock edge, the output Q simply follows (is set equal to) the value present at D at that instant — hence the name “Delay” flip-flop, since the input is delayed by one clock period before appearing at the output. It is widely used in registers because it has no invalid or race condition.

D	Q(next)
0	0
1	1

T (Toggle) Flip-Flop

The T flip-flop has a single input T. When $T = 0$ the output holds its previous state; when $T = 1$ the output toggles on every active clock edge. A T flip-flop can be derived from a JK flip-flop by tying J and K together ($J = K = T$), or from a D flip-flop by feeding back $D = \bar{Q} \oplus T$. T flip-flops are the natural building block for binary counters because toggling implements the divide-by-two behaviour needed at every counter stage.

T	Q(next)	Remark
0	Q (no change)	Hold
1	$\bar{Q}$ (complement)	Toggle

Note: D and T flip-flops are usually built internally from a JK (or Master-Slave JK) flip-flop by tying inputs together — $D = J$, $K = \bar{D}$ and $T = J = K$.

4.1.1 Level Triggering and Edge Triggering

Flip-flops are classified by when they respond to changes at their data inputs, relative to the clock signal.

- Level-triggered (transparent) flip-flops — the output can change at any time the clock is at its active level (HIGH for a positive-level latch, LOW for a negative-level latch). While the clock is active, the circuit is “transparent” and simply follows the input, which can make it sensitive to noise or multiple changes during one pulse.
- Edge-triggered flip-flops — the output changes only at the exact instant of a clock transition (the rising edge, marked $\uparrow$, or the falling edge, marked $\downarrow$). Between edges, the output is completely insensitive to the data input, which prevents race-around and makes circuit timing far more predictable.

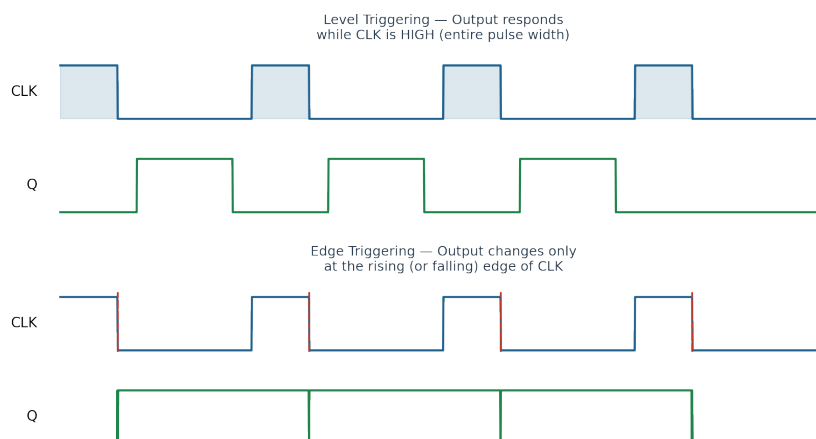


Fig. 4.2 — Level triggering (output follows CLK level) vs. edge triggering (output changes only at the active edge)

Note: Most practical sequential circuits, including all standard counters and registers, use edge-triggered (or master-slave) flip-flops to guarantee predictable, glitch-free operation.

4.1.2 Excitation Tables

An excitation table is the reverse of a truth table: given a required present-state-to-next-state transition ($Q \rightarrow Q(\text{next})$), it lists the input combination that must be applied to force that transition. Excitation tables are essential for designing synchronous sequential circuits, since the designer starts from the desired state transitions and must work backward to the required flip-flop inputs.

Q	Q(next)	S	R	J	K	D	T
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	0	1
1	1	X	0	X	0	1	0

Note: 'X' denotes a don't-care condition — the input may be either 0 or 1 without affecting the required transition, and is used freely during K-map simplification of next-state logic.

Key Takeaways — Flip-Flops

- ✓ RS flip-flop: simplest, but $S = R = 1$ is invalid.
- ✓ JK flip-flop: fixes the invalid state by toggling when $J = K = 1$.
- ✓ Master-Slave JK: cascaded master/slave latches eliminate race-around.
- ✓ D flip-flop: output follows input at the clock edge (no invalid state).
- ✓ T flip-flop: toggles when $T = 1$; the natural element for counters.
- ✓ Edge triggering (vs. level triggering) gives predictable, glitch-free timing.
- ✓ Excitation tables list the inputs needed to achieve a desired $Q \rightarrow Q(\text{next})$ transition.

4.2 Asynchronous and Synchronous Counters

A counter is a sequential circuit built from flip-flops that cycles through a defined sequence of states, usually in step with an input clock. Counters are broadly classified as:

- Asynchronous (ripple) counters — only the first flip-flop receives the external clock; each subsequent flip-flop is clocked by the output of the previous one. State changes “ripple” through the chain, so overall propagation delay grows with the number of bits.
- Synchronous counters — every flip-flop shares the same clock signal, and next-state logic (combinational gates) determines each flip-flop's input. All outputs change simultaneously, giving much faster and more predictable operation for large bit-widths.

4.2.1 Ripple Counter — Circuit, State Diagram and Timing Waveforms

A 4-bit binary ripple-up counter is built from four T (or JK wired as T) flip-flops. Each flip-flop toggles on every falling edge of the previous stage's Q output, so FF0 divides the input clock by 2, FF1 divides by 4, FF2 by 8 and FF3 by 16. The counter therefore steps through the binary sequence 0000, 0001, 0010, ... 1111, then rolls over to 0000.

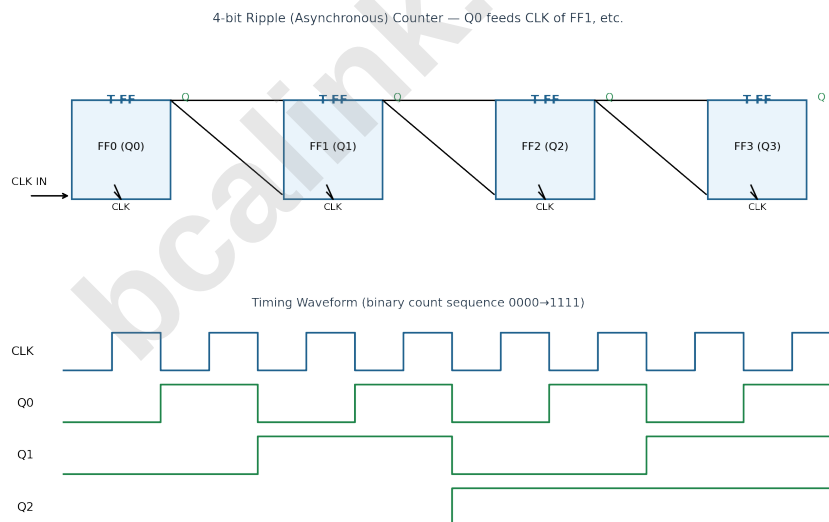


Fig. 4.3 — 4-bit asynchronous ripple counter: circuit (top) and resulting timing waveform (bottom)

Because each stage must wait for the previous stage to settle before it can change, the worst-case propagation delay of an n -bit ripple counter is $n \times t_{pd}$, where t_{pd} is the delay of a single flip-flop. This cumulative delay limits the maximum usable clock frequency, especially for wide counters.

Note: Ripple counters are simple and use minimal hardware (no extra combinational logic), but the ripple delay and momentary invalid output states during transitions make them unsuitable for high-speed applications.

4.2.2 Ring Counter — Circuit, State Diagram and Timing Waveforms

A ring counter is a shift register in which the output of the last flip-flop is fed back to the input of the first, forming a closed loop. A single 1 (or 0) is pre-loaded into one flip-flop, and on each clock pulse it shifts one position around the ring. An n-bit ring counter therefore has exactly n valid states and produces one active output line at a time, making it useful for simple sequencing and timing-signal generation.

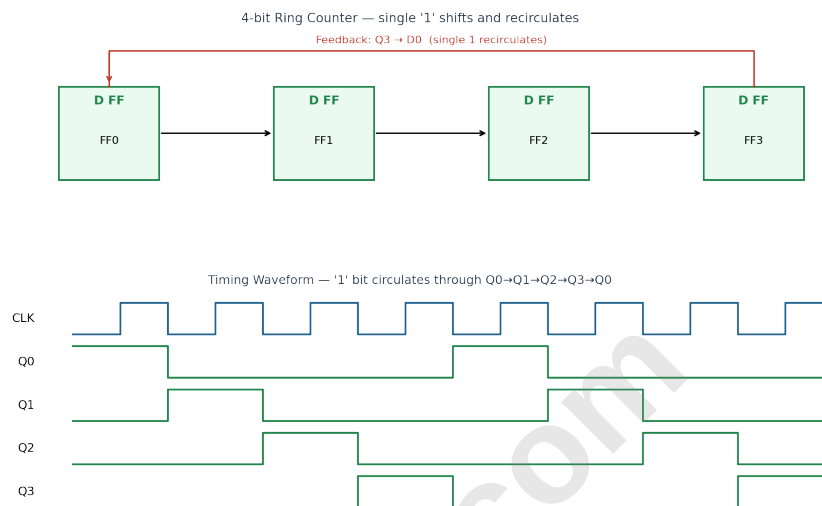


Fig. 4.4 — 4-bit ring counter: circuit with $Q3 \rightarrow D0$ feedback (top) and timing waveform (bottom)

Compared to a binary counter, a ring counter needs no decoding logic to identify “which state” the circuit is in — each flip-flop output directly indicates one state. The trade-off is that it needs one flip-flop per state (n flip-flops for n states), which is far less efficient than a binary counter's $\log_2(n)$ flip-flops.

4.2.3 Modulus-10 Counter — Circuit, State Diagram and Timing Waveforms

A Mod-10 (decade / BCD) counter counts from 0000 through 1001 (decimal 0–9) and then resets back to 0000, instead of continuing to 1111 like a full 4-bit binary counter. It is built from four flip-flops plus a feedback (reset) gate: as soon as the counter reaches state 1010 (decimal 10), the reset logic detects this invalid BCD state and forces all flip-flops back to 0000.

Mod-10 (BCD) Counter State Diagram — states 0000 through 1001, resets to 0000 after count 9

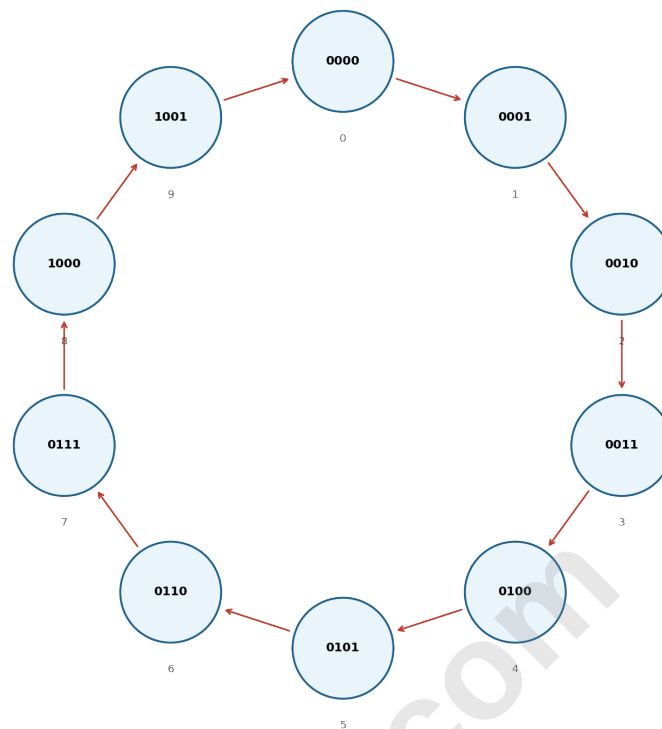


Fig. 4.5 — State diagram of a Mod-10 (BCD) counter: 10 valid states, 0000→1001, then reset

Example 4.2 — Detecting state 1010 for reset

For a 4-bit counter with outputs Q3 Q2 Q1 Q0, decimal 10 = 1010. A 2-input AND gate connected to Q3 and Q1 (the bits that are 1 only at count 10) drives the asynchronous CLEAR input of all four flip-flops, forcing the counter back to 0000 the instant state 1010 is reached.

4.2.4 Modulus Counters (5, 7, 11) — Design Principle, Circuit and State Diagram

A general Mod-N counter is designed using the same reset principle: choose the minimum number of flip-flops m such that $2^m \geq N$, connect them as a binary (or synchronous) counter, and add decoding logic that detects state N and asynchronously resets the counter to state 0.

Modulus (N)	Flip-flops required	Counting range	Reset detects
5	3 ($2^3=8 \geq 5$)	000–100 (0–4)	101 (=5)
7	3 ($2^3=8 \geq 7$)	000–110 (0–6)	111 (=7)
11	4 ($2^4=16 \geq 11$)	0000–1010 (0–10)	1011 (=11)

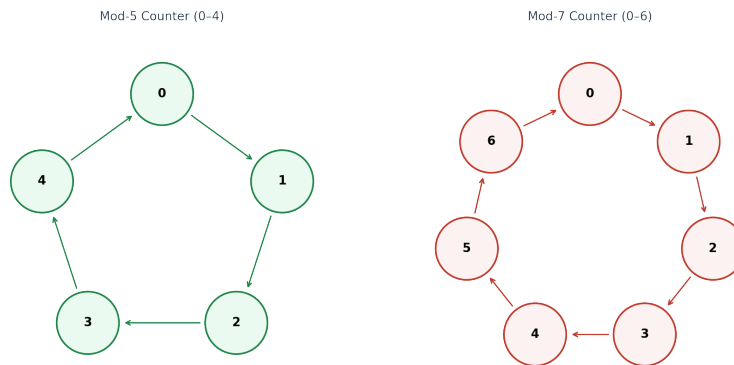


Fig. 4.6 — State diagrams for Mod-5 (left) and Mod-7 (right) counters

Design steps for a Mod-N counter using the reset method:

- Determine $m = \lceil \log_2 N \rceil$ flip-flops.
- Connect the flip-flops as a standard binary (ripple or synchronous) counter.
- Identify the binary code of state N (the first invalid/unwanted state).
- Feed the outputs that are 1 in state N into an AND gate.
- Connect the AND gate output to the asynchronous CLEAR (or PRESET) inputs of all flip-flops, so the counter is forced to 0 the instant state N momentarily appears.

Note: The reset-based design produces a brief, unstable glitch state (state N) that exists only for the propagation delay of the reset gate — this is acceptable in most applications but must be considered in high-speed designs.

4.2.5 Synchronous Design of Counters — Circuit and State Diagrams

In a synchronous counter, all flip-flops are driven by the same clock line, and combinational logic computes each flip-flop's next input directly from the present state — eliminating ripple delay entirely. The standard design procedure is:

1. Draw the required state diagram and derive the state (transition) table.
2. Choose a flip-flop type (commonly JK, since it offers the most don't-care flexibility) and write the excitation table for each bit.
3. Use Karnaugh maps to simplify the next-state input equations (e.g. J_0 , K_0 , J_1 , K_1 , ...) in terms of the present-state variables.
4. Implement the simplified equations with AND/OR gates feeding the J/K (or D) inputs of each flip-flop.
5. Verify the circuit by tracing through every state of the state diagram.

Example 4.3 — Synchronous 3-bit binary up-counter (JK)

For flip-flop FF0 (LSB): $J_0 = K_0 = 1$ (toggles every clock).

For FF1: $J_1 = K_1 = Q_0$ (toggles only when $Q_0 = 1$).

For FF2: $J_2 = K_2 = Q_0 \cdot Q_1$ (toggles only when both lower bits are 1).

This pattern — each stage's toggle condition is the AND of all lower bits — generalises directly to any n-bit synchronous binary counter.

Note: Synchronous counters require more combinational logic than ripple counters, but all outputs change simultaneously with no cumulative delay, allowing much higher maximum clock frequencies.

Key Takeaways — Counters

- ✓ Ripple counters chain flip-flop outputs to the next clock input; simple but slower (cumulative delay).
- ✓ Synchronous counters share one clock and use next-state logic; faster, more complex.
- ✓ Ring counters shift a single active bit around n flip-flops (n states, no decoding needed).
- ✓ Mod-10 (BCD) counters reset at state 1010 to give a 0–9 decimal count.
- ✓ General Mod-N design: use $\lceil \log_2 N \rceil$ flip-flops and reset logic that detects state N.
- ✓ Synchronous design procedure: state diagram → excitation table → K-map → gate-level circuit.

4.3 Application of Counters

Counters are among the most widely used sequential building blocks in digital systems. Two everyday applications illustrate how counting, division and decoding logic combine to produce a practical device.

4.3.1 Digital Watch

A digital watch is fundamentally a chain of counters driven from a highly stable oscillator, typically a 32.768 kHz quartz crystal — chosen because $32,768 = 2^{15}$, so it can be divided down to exactly 1 Hz using 15 cascaded divide-by-2 (T flip-flop) stages.

- Divider chain: the 32.768 kHz crystal output passes through 15 binary (divide-by-2) counter stages to produce a precise 1-pulse-per-second signal.
- Seconds counter: a Mod-60 counter counts the 1 Hz pulses from 0 to 59, then rolls over and sends a carry pulse to the minutes stage.
- Minutes counter: another Mod-60 counter, incremented once per carry pulse from the seconds counter.
- Hours counter: a Mod-12 (or Mod-24) counter, incremented once per carry pulse from the minutes counter.
- Display decoding: BCD-to-seven-segment decoders convert each counter's binary output into the segment patterns shown on the display.

Note: The same divide-by-2¹⁵ and Mod-60/Mod-60/Mod-12 counter cascade structure appears in essentially every quartz clock and watch regardless of manufacturer.

4.3.2 Frequency Counter

A frequency counter measures the frequency of an unknown periodic signal by counting how many pulses of that signal occur during a precisely timed measurement window (the gate time), commonly 1 second.

- Time-base generator: a crystal oscillator and divider chain produce an accurate gate signal, typically active for exactly 1 second.
- Gate (AND) circuit: the unknown input signal is passed through an AND gate that is enabled only during the gate-time window.
- Counter chain: a cascade of decade (Mod-10) counters counts the number of pulses that pass through the gate.
- Display/latch: at the end of the gate period the accumulated count — which equals the signal's frequency in Hz when the gate time is 1 s — is latched and shown on a display, and the counters are then reset for the next measurement cycle.

Example 4.4 — Reading a frequency counter

If the gate time is exactly 1 second and the decade-counter chain accumulates a count of 15,000 pulses during that window, the measured input frequency is 15,000 Hz = 15 kHz.

Key Takeaways — Applications

- ✓ A digital watch divides a 32.768 kHz crystal by 2^{15} to get 1 Hz, then uses Mod-60/Mod-60/Mod-12 counters for seconds, minutes and hours.
- ✓ A frequency counter gates the unknown signal into a decade-counter chain for a precise time window (commonly 1 s) to read frequency directly.

bcalink.com

4.4 Registers

A register is a group of flip-flops used to store multiple bits of data together, typically one flip-flop per bit. A shift register additionally allows the stored data to be shifted, one bit position at a time, either to the left or to the right on each clock pulse. Shift registers are classified by how data enters and leaves the register.

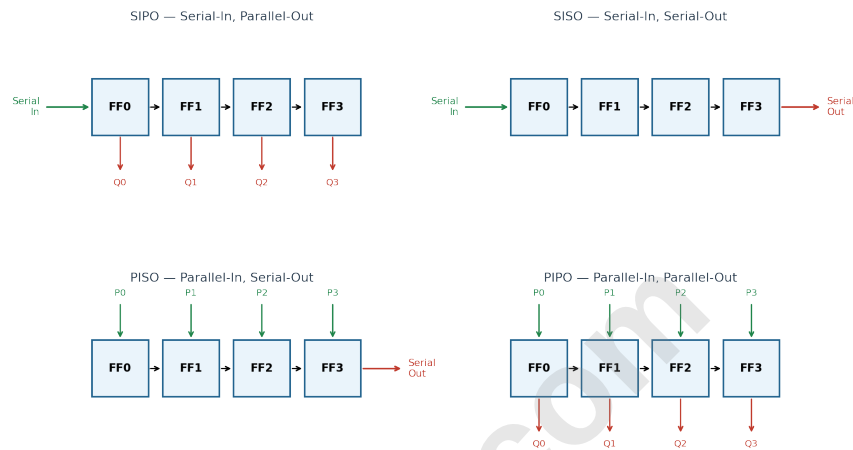


Fig. 4.7 — The four fundamental shift-register configurations

4.4.1 Serial-In, Parallel-Out (SIPO) Register

Data is shifted into the register one bit at a time through a single serial input line. After n clock pulses (for an n -bit register), all n bits are available simultaneously at the parallel outputs $Q0...Q(n-1)$. SIPO registers are commonly used to convert serial data streams (e.g. from a UART) into parallel form for processing.

4.4.2 Serial-In, Serial-Out (SISO) Register

Data enters serially through a single input and, after propagating through all the flip-flop stages, exits serially through a single output. No parallel access to intermediate bits is provided. SISO registers are used purely as fixed-length delay lines, timing elements, or in some simple pseudo-random sequence generators.

4.4.3 Parallel-In, Serial-Out (PISO) Register

All n bits are loaded into the register simultaneously through parallel input lines (using a LOAD control signal), and are then shifted out one bit at a time through a single serial output line on successive clock pulses. PISO registers are the reverse of SIPO and are typically used to convert parallel data into a serial bit stream for transmission, as in a UART transmitter.

4.4.4 Parallel-In, Parallel-Out (PIPO) Register

All n bits are loaded simultaneously through parallel inputs and read out simultaneously through parallel outputs — no serial shifting is involved in getting data in or out (though the flip-flops may still support shifting internally). A PIPO register is essentially used as a simple parallel data buffer or temporary storage/latch element.

Register type	Data in	Data out	Typical use
SIPO	Serial (1 bit/clock)	Parallel (all bits at once)	Serial-to-parallel conversion
SISO	Serial	Serial	Time delay line
PISO	Parallel (all bits at once)	Serial (1 bit/clock)	Parallel-to-serial conversion
PIPO	Parallel	Parallel	Temporary data buffer / latch

4.4.5 Right-Shift and Left-Shift Registers

The direction in which stored data moves distinguishes right-shift from left-shift registers:

- Right-shift register: on each clock pulse, every bit moves one position toward the least-significant (rightmost) flip-flop; $Q_i \leftarrow Q_{i+1}$, and new serial data enters at the most-significant (leftmost) end.
- Left-shift register: on each clock pulse, every bit moves one position toward the most-significant (leftmost) flip-flop; $Q_i \leftarrow Q_{i-1}$, and new serial data enters at the least-significant (rightmost) end.
- A bidirectional shift register combines both capabilities using a direction-control input and multiplexers at each flip-flop's data input, allowing the same hardware to shift either left or right on command.

Example 4.5 — Right-shift operation

A 4-bit register initially holds 1 0 1 1 ($Q_3 Q_2 Q_1 Q_0$). If a new bit '0' is shifted in serially from the left (Q_3 side) on one clock pulse, the new content becomes 0 1 0 1 — each bit has moved one position to the right, and the original rightmost bit (1) is shifted out and lost.

Note: Shifting a binary number left by one bit position is equivalent to multiplying it by 2 (if no bit is lost off the top); shifting right by one bit position is equivalent to integer division by 2 — a property exploited in simple arithmetic and multiplication circuits.

Key Takeaways — Registers

- ✓ A register stores multiple bits; a shift register also moves them left or right each clock pulse.
- ✓ SIPO: serial in, parallel out — used for serial-to-parallel conversion.
- ✓ SISO: serial in, serial out — used as a delay line.
- ✓ PISO: parallel in, serial out — used for parallel-to-serial conversion.
- ✓ PIPO: parallel in, parallel out — used as a simple data buffer/latch.
- ✓ Right-shift moves bits toward the LSB; left-shift moves bits toward the MSB ($\approx \div 2$ and $\times 2$).

— *End of Unit 4* —

bcalink.com