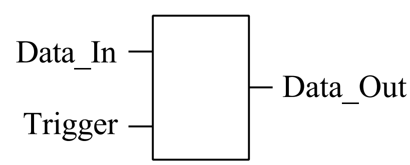
Module for Lab #16: Basic Memory Devices

Revision: November 14, 2004 LAB

Overview

This lab introduces the concept of electronic memory. Memory circuits store the voltage present on an input signal (LHV or LLV) when triggered by a control signal, and they retain the stored voltage until the next assertion of the trigger signal. When the trigger signal is not asserted, memory circuits simply output the most recently stored voltage. At each subsequent assertion of the trigger signal, memory circuits either retain their current state or transition to the opposite voltage (as dictated by the input signal voltage). "Memory" occurs when the output signal remains at a given voltage between trigger signals, even if the input signal changes state.

A memory circuit needs at least two inputs – the data signal to be memorized, and a timing control signal to indicate exactly when the data signal should be memorized. In operation, the data input signal drives the memory circuit's storage node to a '1' or '0' whenever the timing control input is asserted. Once a memory circuit has transitioned to a new state, it can remain there indefinitely until some future input changes direct the memory to a new state. This lab examines basic circuits that can be used to create electronic memory.



Basic Memory Device

Before beginning this lab, you should:

- Be well practiced in the design of various combinational circuits.
- Be familiar with the Xilinx WebPack design tool suite.

After completing this lab, you should:

- Understand the design and function of basic memory circuits.
- Be aware of the potential problems that might arise when memory circuits sample an input signal.
- Be familiar with the various memory devices available in the Xilinx library.

This lab exercise requires:

- A Digilab XCRP board
- A PC running the Xilinx WebPack or ISE CAD tools

Background

Introduction to Memory Circuits

Memory circuits can be segregated into two major groups: those that store data for use in a computer system (such as the RAM in a PC); and those that store information that defines the operating state of a digital system. Memory circuits that are used exclusively to store data in a computer system have become very specialized, and they will be covered in a later lab. This exercise will present memory circuits that are used to store information about the operating state of a digital system.

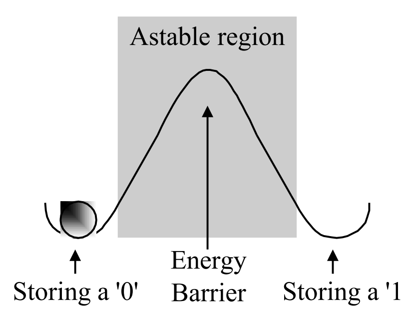
Many electronic devices contain digital systems that use memory circuits to define their operating state. In fact, any electronic device that can create or respond to a sequence of events must contain memory. Examples of such devices include watches and timers, appliance controllers, gaming devices, and computing devices. If a digital system contains N memory devices, and each memory device stores a ‘1’ or a ‘0’, then the system’s operating state can be defined by an N-bit binary number. Further, a digital system with N memory devices must be in one of 2^N states, where each state is uniquely identified by a binary number created from the collective contents of all memory devices in the system.

At any point in time, the binary number stored in its internal memory devices defines the **current state** of a digital system. Inputs that arrive at the digital system may cause the contents of one or more memory devices to change state (from a ‘1’ to a ‘0’ or vice-versa), thereby causing the digital system to change states. Thus, a digital system **state change** or **state transition** occurs whenever the binary number stored in internal memory changes. It is through directed state-to-state transitions that digital systems can create or respond to sequences of events. The next lab will present digital systems that can store and change states according to some algorithm; this lab will examine the circuits that can be used to form memory.

Basic Cells

In digital engineering, we are concerned with **two-state** or **bistable** memory circuits. Bistable circuits have two stable operating states – the state where the output is a logic “1” (or Vdd, or LHV), and the state where the output is a “0” (or GND, or LLV). When a bistable memory circuit is in one of the two stable states, some amount of energy is required to force it out of that state and into the other stable state. During the transition between states, the output signal must move through a region where it is **astable.** Memory circuits are designed so that they cannot stay in the astable state indefinitely – once they enter the astable state, they immediately attempt to regain one of the two stable states.

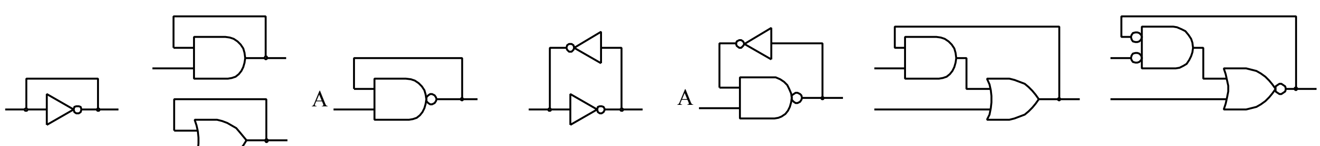
The figure on the right provides an adequate analogy. Here, the ball represents the value stored in memory, and the “hill” represents the astable region that must be crossed before the memory circuit can transition to storing the opposite value. Note that a third potential stable state exists in this analogy - with just the right amount of energy, it would be possible to balance the ball directly on top of the hill. Likewise, memory circuits also have a third potential stable state, midway between the two stable states. When memory circuits transition between their two stable states, it is important to ensure that enough



energy is imparted to the circuit to ensure that the astable region is crossed.

Both the LHV and LLV states in a bistable circuit are easily maintained once they are attained. A control signal that causes the circuit to change states must deliver some minimal amount of energy to move the circuit through the astable state. If the input that causes transition from one stable state to the next delivers the minimum required energy, then the transition happens very quickly. If the control signal delivers less than the minimum required energy, then the circuit returns to its original stable state. But if the input delivers just the wrong amount of energy – enough to start the transition but not quite enough to force it quickly through the astable region – then the circuit can get temporarily "stuck" in the astable region. Memory circuits are designed to minimize this possibility, and to decrease the amount of time that a circuit is likely to remain in the astable state if in fact it gets there (in the analogy, imagine a very pointed summit in the astable region, with very steep slopes). If a memory device were to get stuck in an astable state for too long, its output could oscillate, or stay midway between LHV and LLV, thereby causing the digital system to experience unintended and often unpredictable behavior. A memory device that gets stuck in the astable region is said to be **metastable**, and all memory devices suffer from the possibility of entering a metastable state (more will be said about metastability later).

A memory circuit requires feedback, and any circuit with feedback has memory (to date, we have dealt only with feed-forward, combinational circuits without memory). Any logic circuit can have feedback if an output signal is simply "fed back" and connected to an input. Most feedback circuits will not exhibit useful behavior – they will either be **monostable** (i.e., stuck in an output "1" or "0" state), or they will **oscillate** interminably. Some feedback circuits will be bistable and controllable, and these circuits are candidates for simple memory circuits. Simple feedback circuits are shown below, and they are labeled as controllable/not controllable and bistable/not bistable.



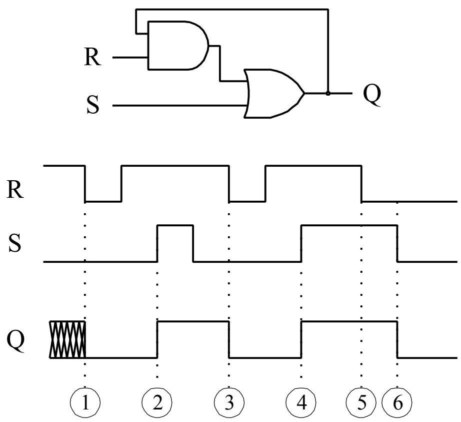
Oscillate | Monostable | Oscillate if A is high | Bistable, not controllable* | Bistable, not controllable | Bistable, controllable | Bistable, controllable

The rightmost two circuits above are both bistable and controllable, and either could be used as a memory element. Timing diagrams for these circuits are shown below.

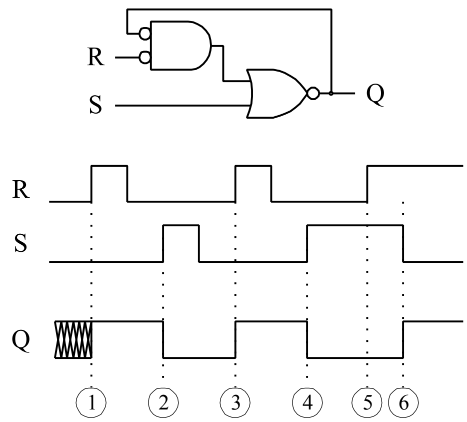
Both circuits below use two inputs named S (for set) and R (for reset), and both use an output named Q (by convention, Q is nearly always used to label the output signal from a memory device). The S input, when asserted, "sets" the output, and the R input "resets" the output. "Setting" an output usually means driving the output to LHV, and resetting usually means driving the output to LLV. However, if a memory device has active low outputs, then setting a device drives the output to LLV.

In the AND/OR circuit on the left below, S must be driven to LHV to drive Q to LHV, and R must be driven to LLV to drive Q to LLV (so S is active high and R is active low). The output Q is set by the positive pulse on S at time 2, and Q remains set until it is reset at time 3. Thus, Q exhibits memory by remaining at LHV after the input S is deasserted, and during the time between point 2 and point 3 the circuit memorized a logic '1'. Likewise, when R is asserted (as a negative pulse), Q is reset to logic '0' and it remains there until it is set sometime in the future, and the circuit memorized a logic '0'.

In the NOR circuit on the right below, S must be driven to LHV to drive Q to LLV, and R must also be driven to LHV to drive Q to LHV (so both S and R are active high). Because the AND/OR circuit requires more transistors, and because its inputs have opposite active levels, it is not used as a memory circuit. The reader is highly encouraged to examine the circuits and timing diagrams below, and ensure that the behaviors shown are well understood.



1. Q is undefined until R is asserted
2. Q is driven to LHV when S is asserted
3. Q is driven to LLV when R is asserted
4. Q is driven to LHV when S is asserted
5. Q remains at LHV when S & R asserted
6. Q is driven to LLV R asserted, S de-asserted



1. Q is undefined until R is asserted
2. Q is driven to LLV when S is asserted
3. Q is driven to LHV when R is asserted
4. Q is driven to LLV when S is asserted
5. Q remains at LLV when S & R asserted
6. Q is driven to LHV R asserted, S de-asserted

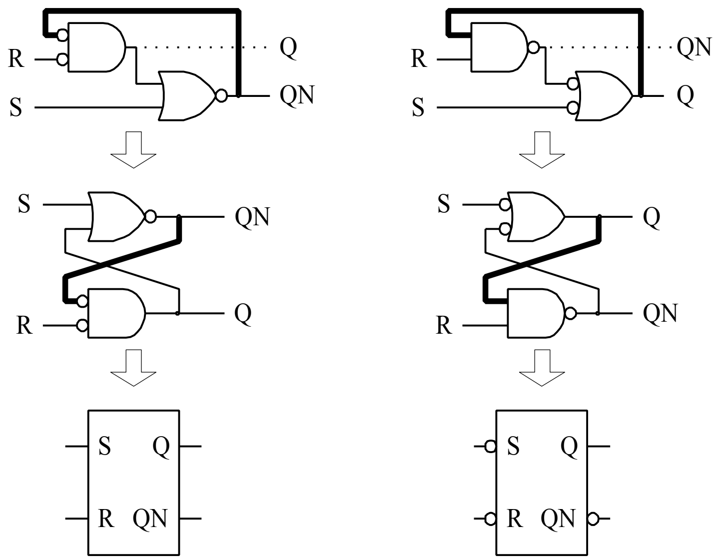
The figure below shows the same NOR circuit and a similar NAND-based circuit. Both of these circuits are frequently used as simple memory circuits, and they are called **basic cells.** The timing diagram for the NAND cell can be easily derived, and it is similar to the NOR diagram shown above. By convention, the basic cell input that drives the output Q to LHV is called SET (or S), and the input that drives Q to LLV is called RESET (or R). The NOR basic cell is said to have asserted-high inputs, because positive pulses on S and R cause memory transitions. The NAND basic cell is said to have asserted-low inputs, because negative pulses on the inputs cause memory transitions.

The NAND and NOR circuits are symmetric, so either input can be labeled S or R. By convention, the output that S drives to LHV is called Q, and the output that S drives to LLV is called QN (and thus the NOR-based circuit above is mislabeled, while the one below is correctly labeled). In the NOR circuit, a LHV on S drives Q to LHV (provided R is at LLV), while in the NAND circuit, a LLV on S drives Q to LHV (regardless of the signal on R). Thus, NOR inputs are active high, and NAND inputs are active low.

In the figure below, the basic cells have been redrawn in the typical cross-coupled topology, with the feedback path emboldened for emphasis. In the NOR basic cell, the Q output is derived from the gate driven directly by R, and so R can determine the output Q regardless of S; this is called a **reset dominant** configuration. In the NAND basic cell, the input S can determine the output regardless of R, and this is a **set-dominant** configuration. The difference between set and reset dominance are evident in the truth table rows where both inputs are asserted. In the reset-dominant NOR cell, Q is forced to LLV when R is asserted (last row), and in the set-dominant NAND cell Q is forced to LHV when S is asserted (first row).

S	R	Q	QN
LV	LV	?	?
LV	HV	LV	HV
HV	LV	HV	LV
HV	HV	LV	LV

NOR centered Basic Cell



S	R	Q	QN
LV	LV	HV	HV
LV	HV	HV	LV
HV	LV	LV	HV
HV	HV	?	?

NAND centered Basic Cell

Examining the truth tables and figure above yields the following observations:

- The middle two rows of the truth tables are similar for both circuits (i.e., both Q and QN are driven opposite from one another when either just S is asserted or just R is asserted).
- When both inputs are asserted, Q and QN are driven to the same logic level (i.e., they are no longer inverses of one another).
- When neither input is asserted, the logic level present on the feedback loop determines the circuit output.

Based on these observations, we can state the following behavioral rules for a basic cell (remembering that SET and RESET are active high for the NOR cell and low for the NAND cell):

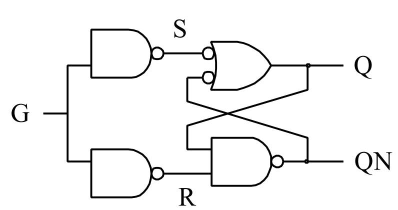
- When just SET is active, Q is driven to LHV and QN is driven to LLV;
- When just RESET is active, Q is driven to LLV and QN is driven to LHV;
- When both SET and RESET are active, Q and QN are both driven to LLV (NOR cell) or LHV (NAND cell);
- When neither SET or RESET are active, the output is determined by the logic value “stored” in the feedback loop.

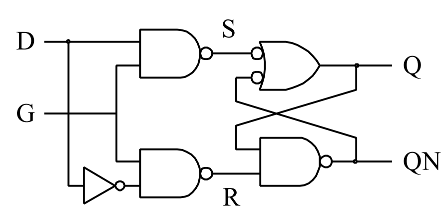
If both inputs to a basic cell are de-asserted at exactly the same time, the feedback loop can become astable. This results from the fact that two different logic levels are introduced into the feedback loop at the same time, and these values “chase” each other around the loop creating a stable oscillation. The oscillation shown in the simulator results from the fact that gate delays can be set to exactly the same value, and inputs can be changed at exactly the same time. In a real circuit, gate delays are not identical and input values cannot change (to the picosecond) at exactly the same time. Thus, oscillations may be seen, but only for a short while. Equally likely is an output that “floats” temporarily between LVH and LLV. Either behavior is termed **metastability**, meaning the memory device output is temporarily not in one of the two stable operating states. Metastable states are highly unlikely in a real circuit, and if they are entered, they are quickly resolved to a stable state. But it is important to note that the possibility of a memory device entering a metastable state can never be eliminated.

Either the NAND or NOR basic cell can be used in practical memory circuits. We will use the NAND cell in the following discussion, but similar circuits could be built with the NOR cell.

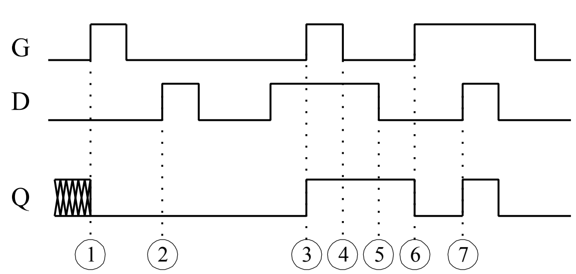
D latch

The basic cell is the most rudimentary memory device, and it is useful in certain situations. But by adding only two logic gates to a basic cell, a much more useful memory device called a D-latch can be created. A D-latch uses a basic cell for a memory element, but it only allows the value stored in memory to be changed (or “programmed”) when a timing control input is asserted. Thus, a D-latch has two inputs – the timing control input and a data input. The timing control input, commonly called “gate”, or “clock”, or “latch enable”, is used to coordinate when new data can be written into the memory element, and conversely, when data cannot be written. In the figure on the left below, observe that when the Gate input is not asserted, S and R are driven to LHV and the output Q is determined by the value stored in the basic cell feedback loop (and so Q is showing the stored logic value). In the figure on the right, observe that when the Gate input is asserted, the D (for Data) input drives S and R to opposite levels, forcing a SET or RESET operation on the basic cell. By combining a timing control input and a data input that forces the basic cell to either SET or RESET, an useful memory device is created. The D-latch is widely used in all sorts of modern digital circuits.



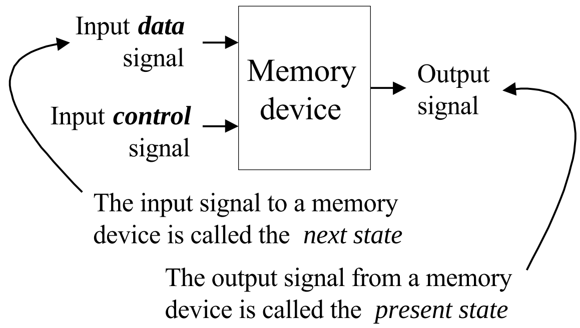


A timing diagram for the D latch is shown below. Note that when the Gate input is asserted, the output Q simply “follows” the input. But when the Gate input is not asserted, the output “remembers” the value present at D at the time the Gate signal was de-asserted.

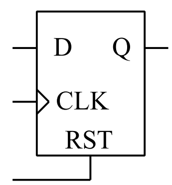


1. Q is undefined until G is asserted; Q gets D's value
2. D is asserted but G is not; Q unchanged
3. D and G are asserted; Q gets D's value
4. G de-asserted; Q memorizes D's value
5. D de-asserted but G also de-asserted; Q unchanged
6. G asserted and Q gets D's value
7. Q follows D while G asserted

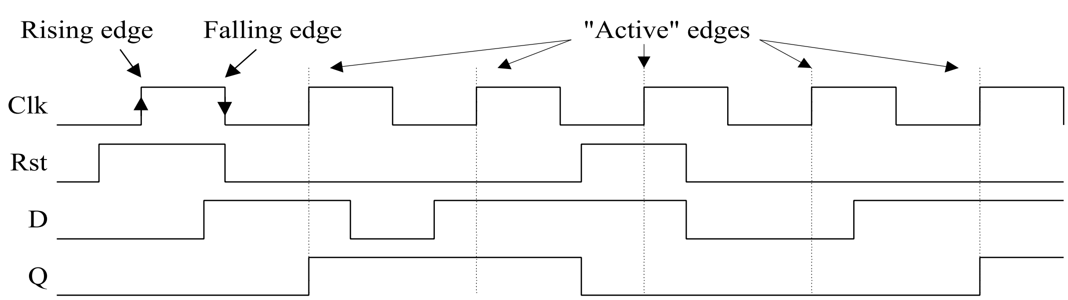
D Flip-Flop

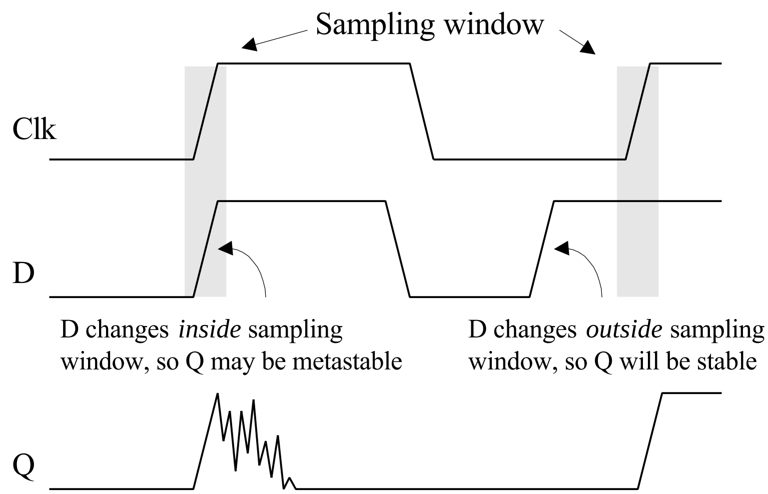


All useful memory devices have at least two inputs – one for the Data signal to be memorized, and a timing control signal to define exactly when the Data signal should be memorized. As shown in the figure, the current output of a memory device is called the **present state**, and the input is called the **next state** because it will define the memory at the next assertion of the timing control input. In a D latch, the present state and next state are the same as long as the timing control input is asserted. A D-flip-flop modifies the function of a D latch in a fundamental and important way: the next state (or D input) can only be written into the memory on the **edge** (or transition) of the timing signal.

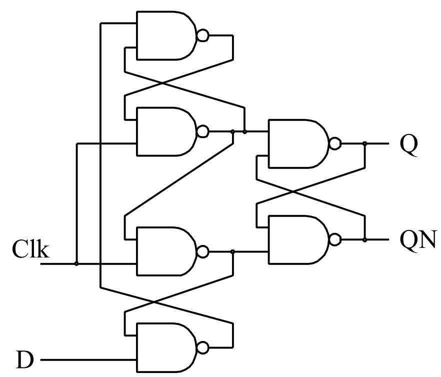


A D-flip flop (DFF) is one of the most fundamental memory devices. It typically has three inputs, including a data input (which must be a '1' or '0'), a timing control input that tells the flip-flop exactly when to "memorize" the data input, and a reset input that can cause the memory to be reset to '0' regardless of the other two inputs. The "D" in DFF arises from the name of the *data* input; thus, the flip-flop may also be called a data flip-flop. The timing control input, called "clock", is used to coordinate when new data can be written into the memory element, and conversely, when data cannot be written. A clock signal is a square wave that regularly repeats at some frequency. A DFF records (or registers) new data whenever an *active* clock edge occurs – the active edge can be either the rising edge or the falling edge. A rising-edge triggered (RET) DFF symbol uses a triangle to show that the flip-flop is edge-triggered; a falling-edge triggered (FET) DFF symbol uses the same triangle, but with a bubble on the outside of the bounding box (just like any other asserted-low input). The timing diagram below illustrates RET DFF behavior. Note that the Q output changes only on the active edge of the clock, and the reset signal forces the output to '0' regardless of the other inputs.





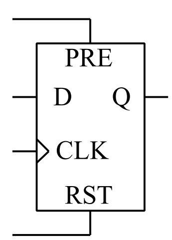
As with the basic cells, a D flip-flop or D latch can enter a metastable state if the data and control inputs are changed at exactly the same time. In a D latch, the data must be stable when the control input is de-asserted. In a DFF, the data input must be stable for a time immediately before and immediately after the clock edge. If the data is not stable at the clock edge, a metastable state may be clocked into the memory element. If this happens, the memory element may not be able to immediately resolve to either low voltage or high voltage, and it may oscillate for a time. Thus, when designing circuits using edge-triggered flip-flops, it is important to ensure the data input is stable for adequate time prior to the clock edge (known as the *setup* time), and for a time after the clock edge (known as the *hold* time). Setup and hold times vary between several tens of picoseconds (for designs inside single IC's) to several nanoseconds (for designs using discrete logic chips).



A schematic for a basic D flip-flop is shown on the right. Several slightly different schematics can be found in various references, but any circuit called a DFF will exhibit the same behavior.

Memory device reset signals

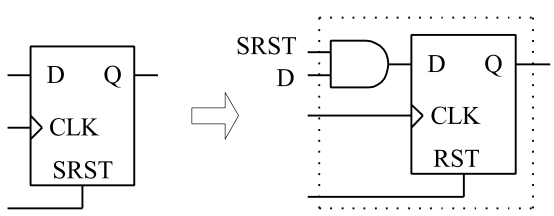
When a memory device is first powered up, it is not possible to predict whether the internal feedback loop will start up storing a LLV, LHV, or metastable state. Thus, it is typical to add a new input signal (or signals) that can force the feedback loop to LHV or LLV. Called **reset** or **preset**, these signals are independent of the CLK or D inputs, and they override all other inputs to drive the stored value to a LLV or LHV respectively. These signals are most useful when a memory device is first initialized after power-on, but they can be used at any time to force the output low or high regardless of the state of the CLK or D signals.



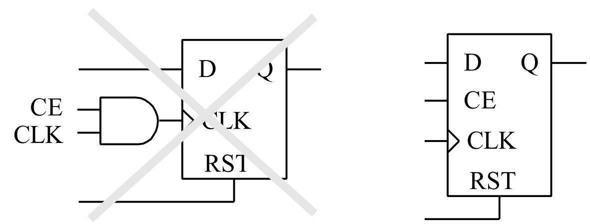
Other inputs to memory devices

In addition to the reset and preset signals, two other signals are often included in memory device circuits. The first, called **clock enable** (or **CE**) can be used to render the memory device either responsive or non-responsive to the CLK signal. In many applications, it is convenient to temporarily disable the clock to a memory device, and it is tempting to do so by running the clock signal through an AND gate with an enable signal driving one side of the gate. This is a poor design technique in any situation (and particularly when designing with FPGA's), because the output of the clock-gating AND gate can glitch, causing unwanted clock pulses to "leak" through. The CE input has been specially designed to disable the clock while avoiding possible glitches.

Another frequently encountered signal in memory devices is a **synchronous reset** that drives the memory device output to LLV on the next rising edge of the clock. The synchronous reset signal simply drives one side of an AND gate inside the memory device (with the other side of the AND gate driven by the D input).



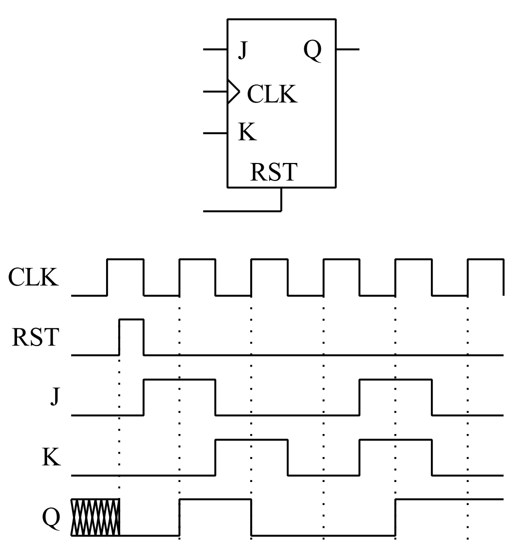
A synchronous reset is just an AND gate on the D input

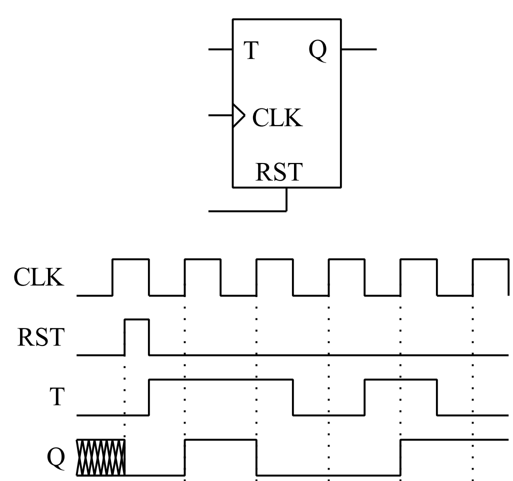


Never gate the clock - use a device with a CE input instead

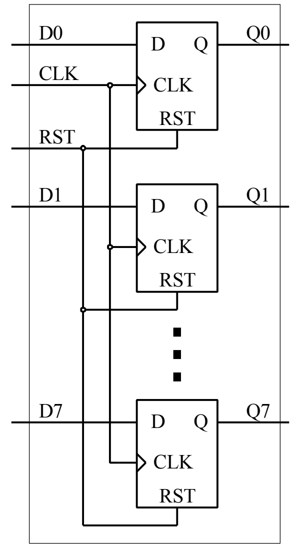
Other flip-flops

The DFF is the simplest and most useful edge-triggered memory device. Its output depends on a Data input and the clock input – at the active clock edge, the device output is driven to match the device's data input. The D-FF can be used in any application that requires a flip-flop. Over the years, other flip-flops have been designed that behave similar to, but not exactly like a DFF. One common device, called a JK flip-flop, uses two inputs to direct state changes (the J input sets the output, and the K input resets the output; if both are asserted, the output toggles between "1" and "0"). Another common device, the T flip-flop, simply toggles it's state between "1" and "0" on each successive clock edge so long as the T input is asserted. These devices were commonly used in older digital systems (especially those built of discrete 7400 logic ICs), but they are rarely encountered in modern designs. Both JK-FF and T-FF can be easily constructed from DFFs or from "first principles" using basic cells. In modern digital design, and particularly in designs destined for FPGAs or other complex logic chips, these other flip-flops offer no advantages and they will not be dealt with further here.





<u>Registers</u>



8-bit D register

A register is simply another name for a memory device. A register is composed of a group of DFF's that share a common clock and reset signal, with each flip-flop having a separate input and separate output. Registers are used when the contents of an entire bus must be memorized at the same time. Common register sizes include 1-bit (which is really just a flip flop), 2-bit, 4-bit, 8-bit, and 16-bit.

As with individual flip-flops, registers may have preset, clock enable, or synchronous reset inputs.

<u>Other memory circuits</u>

Many other circuit topologies that exhibit memory are used in modern digital circuits. For example, dynamic memory structures (like those used in the DRAM memory devices used in PCs) make use of small capacitors to store LHV or LLV temporarily. SRAM structures (like those used in cache RAMs in PCs) use cross-coupled inverters to form a bistable cell (this is the reason for the * symbol next to the inverter circuit in the first figure of this exercise). A feedback loop built of cross-coupled inverters presents a much smaller RAM cell, but it can only be programmed by "overdriving" the output of the feedback resistor. These circuits use large write buffers that can overwhelm the feedback inverter by supplying a much larger current (resistive devices can also be used to prevent large current flow in the feedback inverter). Non-volatile memory devices (such as the FLASH BIOS ROM in PCs) use floating gates to store memory bits. Together, these "other" memory circuits make up the vast majority of memory devices in use today. The basic cell and flip-flop circuits shown here are conceptually simple, but they are not that common in modern digital design. These other memory circuits will be covered in more detail in later exercises.