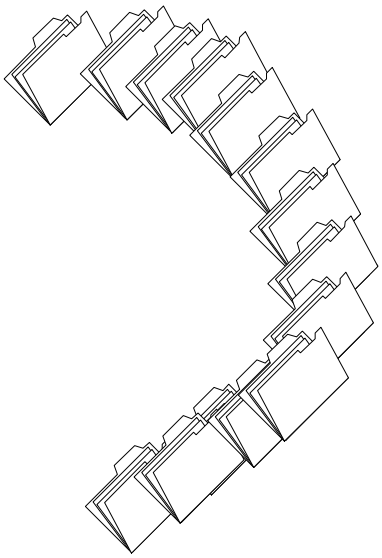
SEQUENTIAL CIRCUITS

Circuits With Storage



Sequential Circuits

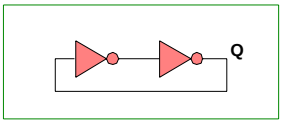
These are the Interesting Circuits

- They can remember.
- The elements that remember are latches and flip-flops.
- RAM is nothing but a very compact collection of latches.

- They also allow many more opportunities to mess up.
- If you send a fast glitch into a latch it may remember the glitch. This is not what you want.
- The clock is an organizational method that, if correctly used, solves some of the problems with circuits which remember.

Latches and Flip-Flops

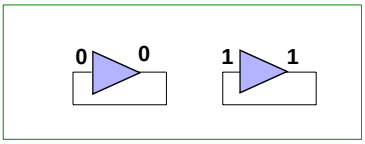
The Basic Digital Latch



Two inverters connected in a loop.

One noninverting gate in a loop

- In this gate: output level = input level.
- But it has power gain;
  The output can supply input power losses.



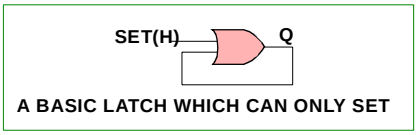
These latches will store whatever bit comes up on power up.
They will remember that bit until the power is shut off.

There is no way to set or reset these latches

Adding SET and RESET to the Basic Latch

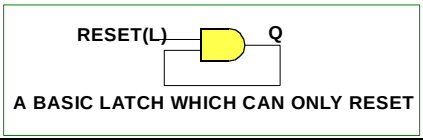
Adding Set

Replace the noninverting gate with an OR gate.



Adding Reset

Replacing the noninverting gate with an AND.t



Sequential Circuits

Latches and Flip-Flops

Feedback

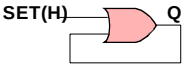
Feedback between gates will do one of two things:

1. Oscillate, if the number of inverters in the feedback loop is odd.
2. Remember, if the number of inverters in the feedback loop is even.
   Zero is an even number.

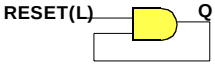
Changing What is stored

A circuit that remembers continuously cannot change what it remembers.
To change what is remembered one must break the feedback loop and interrupt the storage.

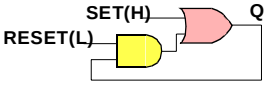
The Basic Latch Converted to a Reset-Set latch



A BASIC LATCH WHICH CAN ONLY SET

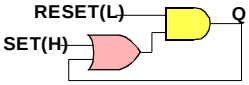


A BASIC LATCH WHICH CAN ONLY RESET



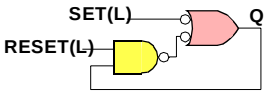
ADDING A RESET TO THE SET LATCH.

Set dominates Reset because
Set acts on Q directly



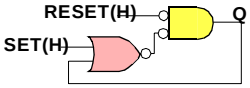
ADDING A SET TO THE RESET LATCH.

Reset dominates Set because
Reset acts on Q directly



THE TWO-NAND-GATE SET-RESET LATCH.

The inputs are asserted-low
Set dominates Reset when both are applied at once.



THE TWO-NOR-GATE RESET-SET LATCH.

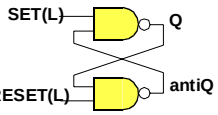
The inputs are asserted-high
Reset dominates set when both are applied at once.

The Reset-Set Latch

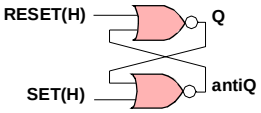
The And-Or Circuit

- R-S latches are made from cross-coupled NANDs or cross-coupled NORs. However drawing them that way is the hardest way to understand them.
- The AND-OR circuit shows easily:
  a. which has asserted low inputs.
  b. which is Set dominant and which is Reset dominant.
- However it does not show the so called “not Q” output.

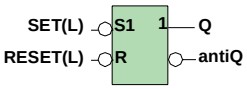
Common Symbols for the RS latches.



A COMMON, BUT LESS CLEAR, WAY OF SHOWING THE TWO-NAND-GATE LATCH.

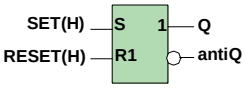


A COMMON, BUT LESS CLEAR, WAY OF SHOWIN THE TWO-NOR-GATE LATCH.



THE SYMBOL FOR THE TWO-NAND-GATE LATCH.

The "1" shows that the Set dominates the Reset when both are applied at once.

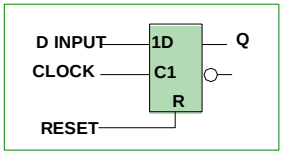


THE SYMBOL FOR THE TWO-NOR-GATE LATCH

The "1" shows that the Reset dominates set when both are applied at once.

Application

- RS latches have limited use in VLSI design.
- D latches and flip-flops are the common storage devices.
- D flip-flops usually have an auxiliary Reset (Clear) which uses the reset circuit like that shown above.



Symbols for RS Latches

The IEEE Symbols

These use a number to show set (reset) dominance. The number joins the Q output with the dominant input.

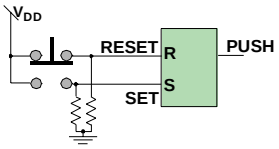
The antiQ pin

For latches two outputs can be obtained. One is the opposite of the other unless both *set* and *reset* are asserted simultaneously. For this reason the author likes to call the second output *antiQ* rather than $\overline{Q}$

Applications of RS Latches

Debouncing Switches

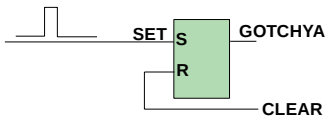
Use a switch that has an upper and lower set of contacts, which are never both contacting at the same time. If the switch is pushed, then the switch may bounce on the set contacts. But unless it bounces back so far it touches the reset contact the latch output will be stable.



Fast Pulse Catcher

To capture and hold a "glitch" on a line which may be much faster than the clock.

This property of capturing glitches is one reason why RS latches are not widely used.



D Type Latches and Flip-Flops

D-Latch Means *Delay*

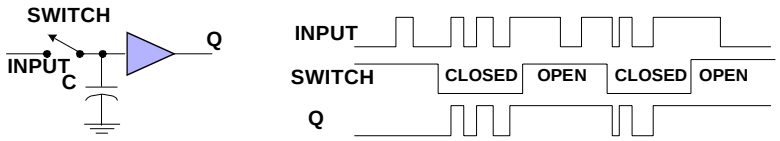
D latches/flip-flops get their name from "delay." D flip-flop delay data for one clock cycle.

The simple dynamic D latch

The simplest D latch stores a 0 or 1 on a capacitor.

1. **Switch closed; the capacitor charges to the input signal level.**
2. **Switch opens, the capacitor remembers the old signal level.**

Logic which remembers using a capacitor is called *dynamic logic*.



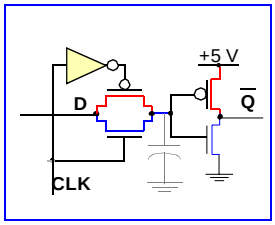
Possible construction of dynamic latch

Normally the switch is operated by a clock signal.

The capacitor must not discharge between clock cycles.

- **The gate must not discharge the capacitor.**
- **Gate made from MOS transistors have the required high input impedance.**

The switch is an MOS transmission gate.



D Latches

The Simple Dynamic D Latch

- The simplest D latch is contains a switch, a capacitor and a gate.
- It functions as follows:

1. The switch is closed, the capacitor charges to the input signal level.
2. Then when the switch opens, the capacitor stores the old signal level.
3. When the switch closes again the output will change to the new input signal level.

Logic which depend on charge stored on a capacitor is called dynamic logic.

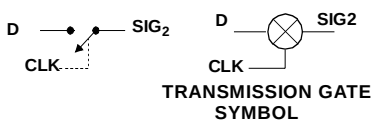
Construction of the Latch

The switch is a transmission gate.

Recall the PMOS transistor had a low resistance for D high.
The NMOS transistor had a low resistance when D was low.
They both turned on and off together.

The inverter has a very high input impedance.

The gates are a capacitor with gigaohms resistance to ground.
They will not discharge the capacitance.
They are part of the storage capacitance.



The Static D Latch

A static latch uses feed-back to remember, not a capacitor.

A static latch can remember as long as gate power is supplied.

The capacitor is only used to remember the state during the switch change over.

The Transparent Latch

The D Latch is often called a *transparent* latch.

Operation of the Static D (Transparent) Latch.

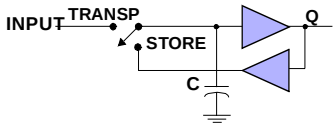
Switch is in the TRANSP position:

- The INPUT signal passes directly to the Q output the latch is said to be *transparent*.

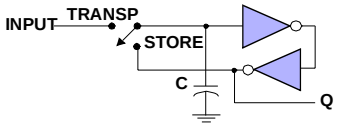
Switch is in the STORE position:

- The last INPUT, just before the switch moved, determines Q.

Adding feedback to convert a dynamic latch to a static latch.



THEORETICAL D LATCH.



A D LATCH BUILT WITH INVERTERS

The Static D Latch

A static latch uses feed-back to remember, rather than depending on the charge on a capacitor.

A static latch can remember as long as power is supplied to its gates.

The dynamic D latch is made into a static latch, by adding another contact to the switch and another noninverting gate.

The capacitor is only used to remember the state during the time the switch is going from one contact to the other.

Operation of the Static D Latch.

When the switch is in the TRANSP position, the INPUT signal passes directly through to the Q output as though the latch were transparent.

When the switch is in the STORE position, the last INPUT, just before the switch was moved, determines Q.

The switch switches between the two contacts, and never rests in the “half way between” position, shown.

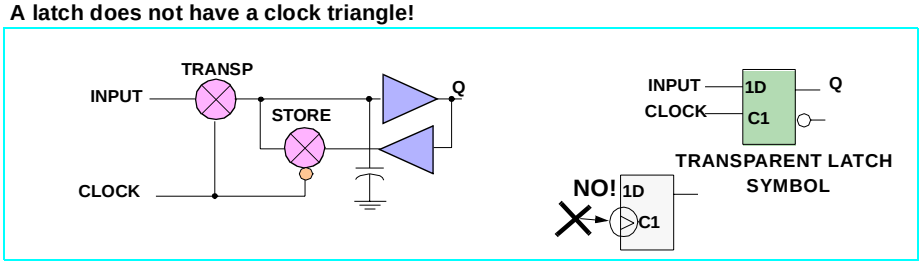
The D Latch With CMOS Switches

This D latch has CMOS switches controlled by the clock.

Normally a switch is on when the clock is high.

The is on when the clock is low.

The D latch symbol is given on the right.
A latch does not have a clock triangle!

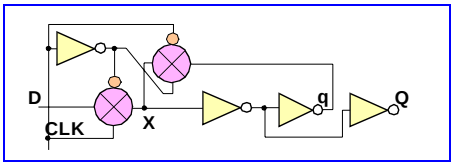


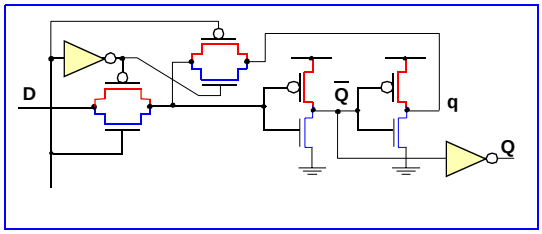
The Complete CMOS Latch

12 transistor latch

Output buffer isolates feedback.

Local clock inversion can be shared?





CMOS Transparent Latch

Two diagrams are shown. One shows most of the transistors,
The other clumps all transistors into inverters or transmission gates.

Size

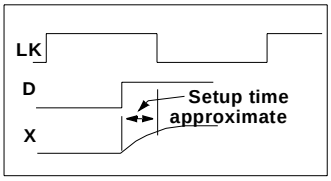
The complete latch uses about 12 transistors.

An output buffer allows a Q, instead of a $\overline{Q}$ output. It also isolates the load from the feedback loop. Without it, a large load capacitance could slow down the latch. It would give a long *setup time* (see below).

The local inverter on the clock could in theory be shared with nearby latches but this is likely not practical since the latch would probably be built into a standard cell.

Setup Time

Setup time is the time the signal D must be stable before the clock goes low. Node D must have time to charge node X through the transmission gate. The delay shown is barely enough, since X is only 80% risen.



The extra resistance of the transmission gate slows down the transfer of charge from D to X.

Also consider the output buffer. Without it any load capacitance would be connected to q instead of Q. With the clock low, there is a connection between q and X; charge can flow either way.

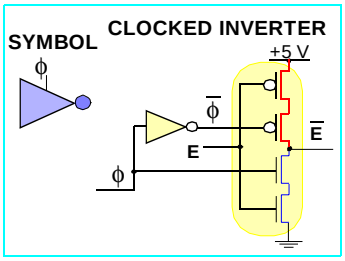
Consider:

1. D rises late. X has time to rise, but q, because of its big capacitance, is still low.
2. The clock goes low connecting q to X. Then charge flows from X to q; it should flow q to X. Then the point X goes low, and the latch "unlatches."

The Clocked-Inverter CMOS Latch

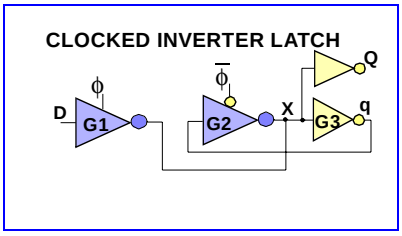
The Clocked Inverter

- Hybrid between a gate and a transmission gate.
- $\overline{E}$ floats when clock ϕ is low.
  E inverts input E when clock ϕ is high.
- Symbol shows clock ϕ coming in the top.



A Clocked-Inverter Latch

- A 14 transistor latch (compare with 12 before)



Operation

- With ϕ high, G1 is on, latch is transparent,
  The D signal travels D→X→q
- With ϕ low, G2 is on, latch stores
  X→q→X forms a noninverting feedback loop.

Pro-Con (compared to transmission gate latch)

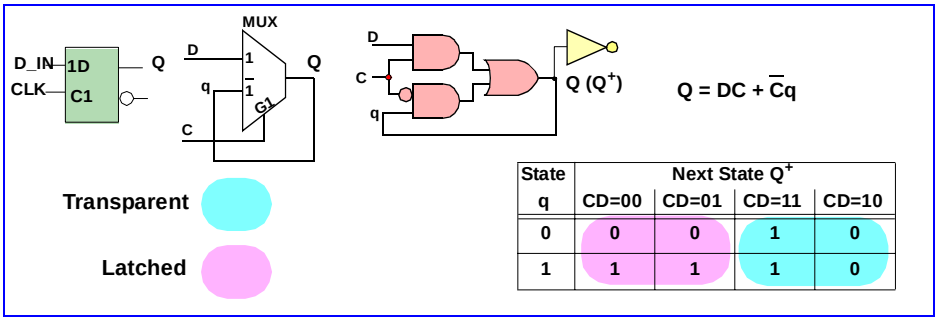
- It also needs a buffer to keep load capacitance from changing setup time.
- Its input buffer reduces interaction with source resistance.
  This keeps timing simulation uncomplicated.
- It eliminates transmission gate testing problems.
  A transistor failure will cause a fault, not a slow, circuit.

Extending the above discussion, deduce that the setup time for the buffered latch is the time from the last change in D to the time q, not just X, is over half charged.

The MUX D-Latch

An Alternate Transparent D-Latch

This form of D-latch (transparent latch) is just a MUX.
When C=1, D feeds through to Q.
When C=0, the previous Q is stored in a noninverting loop.



State	Next State Q^+			
q	CD=00	CD=01	CD=11	CD=10
0	0	0	1	0
1	1	1	1	0

- **With complex gates, this has 12 transistors, 14 with output buffer.**
- **No transmission gates to test.**
- **Input is buffered. Setup time is independent of the source's output impedance.**
- **It still needs an output buffer to isolate setup time from load**
- **If C and $\overline{C}$ have different delays, a glitch is generated which may get latched.**

The Clocked-Inverter Latch[1]

Comparison to Transmission Gate Latch

Size

- This latch has two more transistors than the transmission-gate latch, 14 total with an output buffer. However the layout is simpler and this has the same or less area than the transmission-gate latch.
- The input is not through a transmission gate. The buffer isolates internal latch delays from the impedance of the driving circuit. The transmission-gate delay is hard for a simulator to deal with[2].
- This latch also needs an output buffer. If point X is connected to the load, the time it takes X to charge will vary with load.
  However the time it takes X to charge effects the *setup time*.
- Having a latches load effect the setup time is a horrible thing for a circuit simulator.
- When the latches are made into flip-flops, some of the loads (sources) become known, and hence the an extra buffer may not be needed.

Testing

- Recall that if one of the transistors opened in a transmission gate, the circuit would work but more slowly. At speed tests are difficult.
- There are no transmission gates in the clocked-inverter circuit.

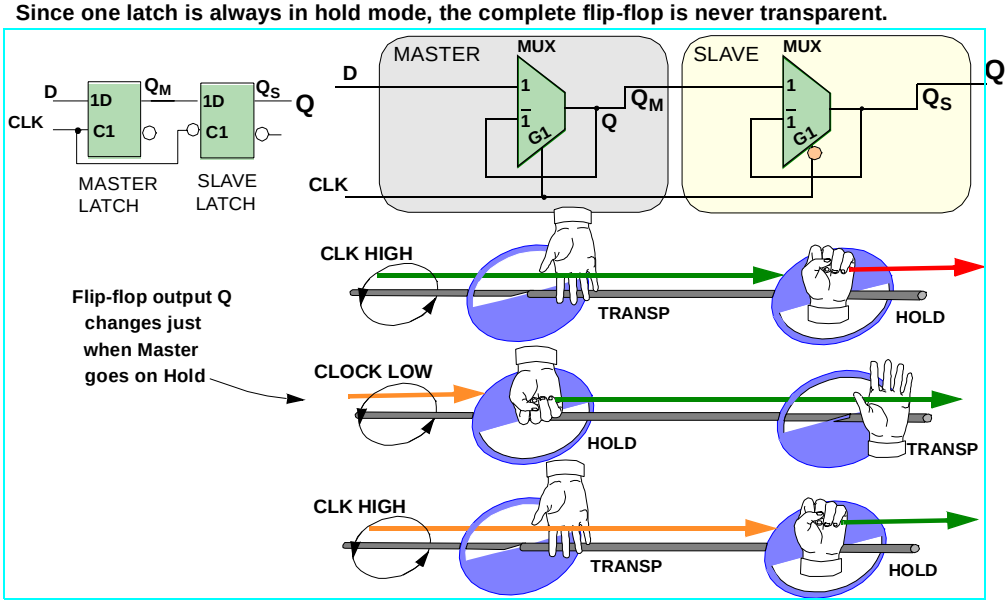
1. Wayne Wolf, *Modern VLSI Design, A Systems Approach*, Prentice Hall, 1994. pp. 156-157.
2. Weste and Eshraghian, *Principles of VLSI CMOS Design, ed. 2*, Addison Wesley, 1993, p 389.

Operation of the master-slave D flip-flop

Two D latches in series and clock one from CLK and one from $\overline{CLK}$.

When the MASTER is transparent the SLAVE is holding.
When the MASTER is holding the SLAVE is transparent.

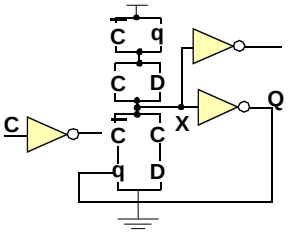
Since one latch is always in hold mode, the complete flip-flop is never transparent.



The MUX D-Latch

Construction

This is effectively the same size as the other two, 12 or 14 gates.



The Glitch Problem

Suppose the D input was high, ready to transfer a 1 to Q (transfer 0 to X) when the clock goes low.

Now suppose C→ $\overline{C}$ was delayed in the inverter so C and $\overline{C}$ are both 0 at the same time.

This would pull point X high for an instant, long enough to store an erroneous 0 in Q.

The glitch problem can be cured by changing the latch equation to

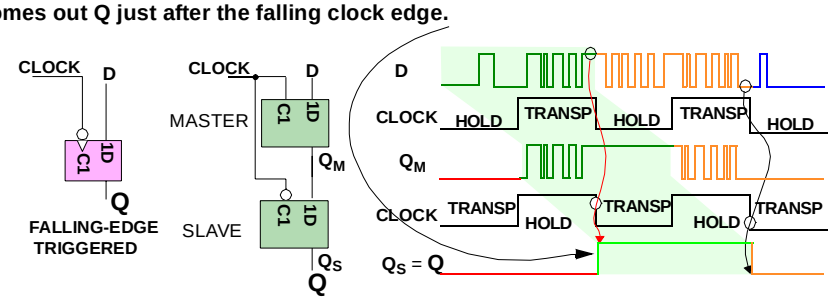
$$Q = DC + \overline{C}q + Dq$$

1.• PROBLEM

Add gate(s) to the MUX-D latch to clear (reset) the output.

It is *Edge-Triggered*

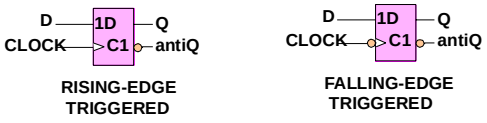
- **The data that enters D just before the clock falls comes out Q just after the falling clock edge.**



- **All master-slave D flip-flops are edge-triggered but not all edge-triggered D flip-flops are master-slave.**

D Flip-Flop Symbol

The symbols for the master-slave D flip-flops. Note the little triangles ▷C1.



The Master-Slave Edge-Triggered D Flip-Flop

When one hand (latch) holds the signal, the other is transparent.

Both hands are never open at once.
The complete flip-flop is never transparent.

Classification of Latches and Flip-flops

Classification by Inputs (Classification by Letters)

1. RS (Reset-set) Latches
2. D (Delay) type latches and flip-flop. They may have R-S inputs also to allow clearing on start-up.
3. JK flip-flops which also may have R-S inputs. The JK flip-flops organization was very good when logic chips had 2 flip-flops per package. They are much less useful for large modern ICs.
4. T (Toggle) flip-flops which change output every clock cycle. Usually they have an enable input to turn on and off the toggling.

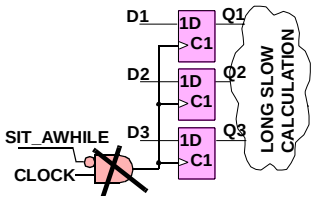
The Enabled D Flip-Flop

Why We Need An Enable

Often need to hold data for several clock cycles.

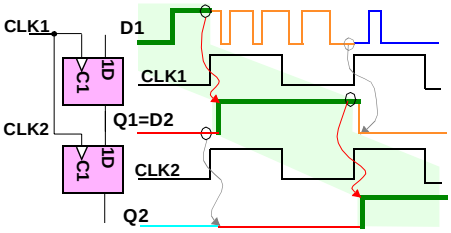
Garbage data on the inputs which we don't want.

So why not just turn off the clock?

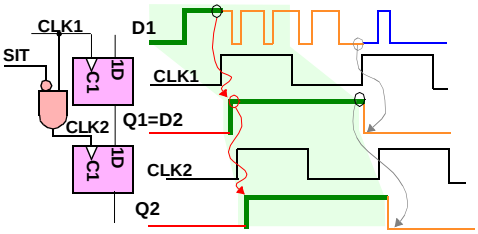


Placing Gates in the Clock Lead Causes *Clock Skew*

Example using a two flip-flop shift register.



An orderly progression
D1 shifts 1 ff on each clock edge.



Gate delays CLK2
CLK2 delay causes 2 shifts on 1 edge.

Other reasons for not gating the clock

- It can cause false clock edges if SIT changes when CLK=1.
- Scan based testing has problems with gates in clock lines.

Classification of Latches and Flip-flops (cont.)

Classification by Clocking Method

Latches

a. The ***unclocked latch***. An RS latch with no other inputs is of this type. They have no clock input. They respond to Set or Reset signals only. ***Unclocked*** or ***asynchronous*** inputs on flip-flops are used to reset or clear the flip-flop.

b. ***Gated latches***, or ***level-sensitive latches***. These latches accept data on one clock half-cycle (usually high) and they store data on the opposite (usually low) half-cycle. An example the D type level-sensitive (or transparent) latches.

Flip-flops

c. ***Edge-triggered*** flip-flops and ***master-slave*** flip-flops. These are the circuits that are properly called flip-flops. Edge-triggered flip-flops accept new data only at a clock edge. Master-slave flip-flops are edge triggered but some JK and RS master-slave flip-flops will also capture a fast pulse at certain other times. Such JK and RS flip-flops are said to be *one's catching* and should be avoided[1].

d. ***Enabled*** latches/flip-flops. These are D, T or JK flip-flops with an extra ENABLE input. D and JK flip-flops will not accept new data until this ENABLE signal is asserted. T flip-flops will not toggle if ENABLE is not asserted.

[1] The master-slave JK flip-flop built out of NAND gates that appears in most textbooks is a one's catching circuit. One's catching excludes a flip-flop from being called edge triggered. The master-slave RS flip-flop is little used.

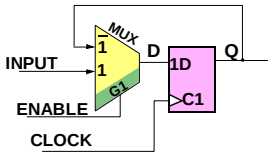
The Enabled D Flip-Flop (cont.)

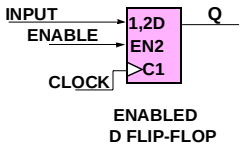
It is a simple way to hold the previous Q clock gating problems.

It switches the flip-flop D input between the old Q, and the new input.

Operation

- **When ENABLE is high the INPUT is fed into the D flip-flop.**
- **When ENABLE is low Q reloads the flip-flop from its own output.**





ENABLED
D FLIP-FLOP

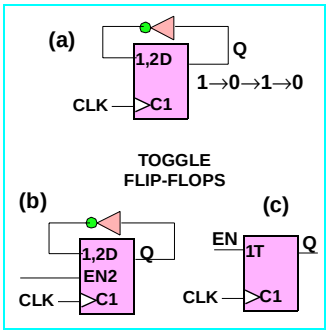
The Toggle Flip-Flop with Enable

The Toggle Flip-Flop

a. **This toggles 1→0→1→0→. . . on every active clock edge.**

b. **An enable makes it more useful.**

c. **The enable is usually called the T input.**



Enabled D Flip-Flop

The Reason For the Enable

The edge-triggered D flip-flop clocks in new data every cycle.
Often one wants to hold the data in the flip flops for a few clock cycles.
Suppose good data is in the flip-flops which must be held for 32 cycles while its square root is calculated.
During the last 31 cycles the D input data is garbage and is to be ignored.

Gating the Clock

One solution is to turn off the clock for the last 31 cycles. This is a dangerous solution.

1. The extra delay in the clock line can cause circuits to skip cycles.
2. If SIT pulses when the clock is high, CLK2 will follow the pulse and cause an extra edge in the ff.
   One must only change SIT when CLK1 is low. This makes designing harder.
3. Scan testing is a method where the flip-flops are extensively used for testing. The test programs demand that no gates be put in clock lines.

Poor practice

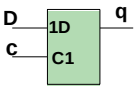
Gating the clock is like using “go to” in programming. One can do it, and one can make proper designs when one does it. However one must make a lot more effort to be sure the design will work under all conditions.

Verilog Latches and Flip-Flops

Verilog Transparent Latch

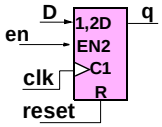
```
// D_latch
  wire c, D;
  reg    q;          // q stores any values put in it until it is changed .
     always @(c or D)   // start of a procedure.
       begin            // begin and end frame the procedure.
         if (c) q=D;
       end
```

Procedure



Verilog D Flip-Flops

```
// D_ff
  wire    clk, D, en, reset;
  reg q;  // q stores
     always @(posedge clk or posedge reset)
       if (reset)  q = 1'b0;
         else if (en) q = D;
```



Enabled D flip-flop

Application

The enable allows the flip-flop to hold data stable for as long as desired without placing gates in the clock lead.

Symbol

The symbol for the enabled D flip-flop is shown on the right.

It uses the number 2 to virtually connect the EN input to the D input

The "1,2D" means both signals 1 and 2, i.e. both ENABLE and the CLOCK edge, are needed for the flip-flop to accept a new D input

Toggle Flip-Flops (T Flip-Flops)

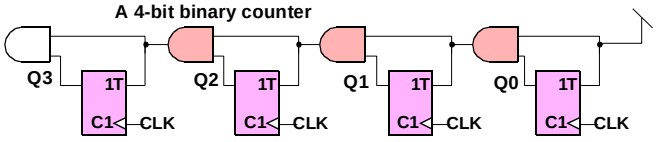
These seldom appear as standard component; they are made from D flip-flops.

They are very useful conceptually.

For example a binary counter is easily made by cascading a basic unit made of a T ff and an AND gate

Q3	Q2	Q1	Q0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
...	...	...	...

A 4-bit binary counter



Shift Registers and Counters

Shift Registers

Standard Shift Registers

Mobius (Johnson) Counters

Linear Feedback Shift Registers

Counters

Ripple

Binary, With Toggle FF

Up-Down Binary

Gray Code

Other Sequential Circuits

Bit-Serial Adder

Verilog Latches

D_latch

- `Reg` variable are like C variables. Anything assigned to them is remembered. Other Verilog variables are merely names given to connections. They are driven by a gate output.
- Procedures, which are like C procedures, start with `always` or `initial`.
  They are framed by `begin` and `end`.
  `If`, `case`, `while`, `for` and some other statements must be put in procedures.
- The guard or trigger statement `@(c or D)` means execute this procedure only if `C` or `D` change
- `If (c) q=D;` says nothing about what `q` is on start up.
  Here q will be the unknown value, X, until `clk=1`.

D_ff

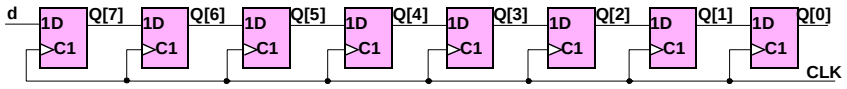
- The procedure here is only one statement, so the begin and end can be omitted.
- The guard statement `@(posedge clk ...` means execute the procedure only on a rising clock edge. There is also `negedge`. Posedge and negedge tell the synthesizer to use edge-triggered flip-flops.
- The `reset` and `clk` interact as follows:
  Any rising `reset` edge will clear the flip-flop.
  If reset is a steady one, a rising `clk` will activate the procedure, but the high `reset` will set `q = 1b0`.
- q can change only if `reset` =0, `en`=1, and `clk` just went high.

2.• PROBLEM

Write the Verilog code for a falling-edge-triggered enabled toggle flip-flop.

Shift Registers

Shifts With Registers



reg [7:0] Q;

- Can multiply by 2^m by left shifts. (`Q<<m`);
- Can divide by 2^m by right shifts. (`Q>>m`);
  More compact than a barrel shifter, but slower.
- Useful for parallel-to-serial and serial-to-parallel conversion.

Verilog Shift Register

```
module shftreg1(Q, d, clk);
  input   clk,  d;
  output  [7:0] Q;
  reg     [7:0] Q;     // This says Q is storage.
      always @(posedge clk)   // This says it is edge-triggered storage.
        begin
        Q<=Q>>1; Q[7]<=d; //Shift Q right 1 place; fill the empty place with d.
        end
endmodule
```


Verilog latches and flip-flops (cont.)

Asynchronous Reset Compilier Directive

Synopsys may have problems synthesizing the desired circuits for asynchronous reset, particularly for latches.

There are directive comments which start " // `synopsys`." These are ignored by the simulator, but give directions about the proper way to sythesize the circuit.

Inserting:

```
// synopsys async_set_reset "clear"
```

in a latch module will ensure a latch with an asynchronous reset, controlled by signal `clear`, is generated.

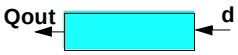
Shift Registers

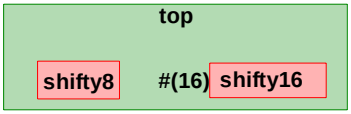
Alternate Code for a Shift Register

Declare `sReg` **an (n-1) bit register,**
then use `{Qout, sReg} = {sReg, d}`

Alternate code with parameters

```
module shftreg2(Qout, d, clk);
  parameter n=8        // lets us change to a 16 bit machine easily.
  input   clk, d;
  output  Qout; reg  Qout;
  reg [n-1:0]  sReg;
      always @(posedge clk)
      {Qout, sReg} <= {sReg, d};
endmodule //shftreg2
```





Overriding parameters from above

```
module top
 //parameter n will be overwritten in instantiation shifty16
. . .
shftreg2 #(16) shifty16(q, d, clk); // Instantiate a shift reg.
          // First parameter in shifty16 overridden to 16.
shftreg2  shifty8(q, d, clk);  // This implementation will go back to 8 bits.
. . .
```


Shift Registers

The Drawing

This was drawn as though storing a number, so theleast-significant bit is on the right.
This means the input is on the left, which is unconventional.

Clock Skew

The clock is shown coming in at the opposite end from the data. This is the best way to layout shift registers.

Normally the flip-flop clocks in data that has been resting for nearly half a clock cycle.
It is when the data changes before the clock that one skips cycles see ."Placing Gates in the Clock Lead Causes Clock Skew," p. 199. However running the data in the opposite direction from the clock means any delay in the wires will skew the clock so the a flip-flops clock earlier than the incoming data.

The Shift Operator

The >> and << operators shift right and left.

`Q >> m` cause `Q` to be shifted right `m` places.

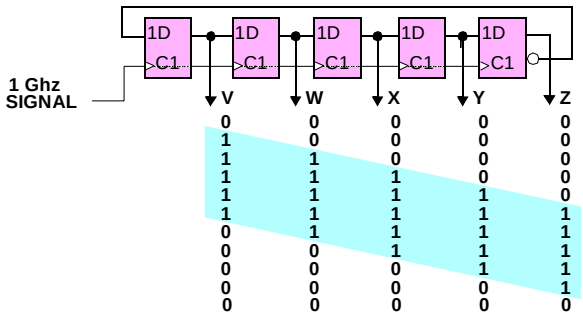
Unless Q is an integer or real variable, the shifts will zero fill from the appropriate end.
Real and integer variables do sign extension.

Mobius (Johnson) Counters

A rotating shift register with the output bit inverted before it is fedback.

- **Johnson or Mobius counters change only one output bit at a time.**
  Reading the counter while it is changing causes no error.
  You get either the last reading or the next.
- **The counter has only to 2N different counts, not 2^N.**
  Not bad for N=3 (2N=6, 2^N=8); bad for N=10 (2N=20, 2^N=1024).
- **The counter is very fast because there are no gates between the flip-flops.**

A Johnson counter. The clock line is shown running behind the flipflops



Alternate Code for a Shift Register

There are several concepts here:

1. The use of concatenate operations in the code.
2. The use of parameters.
3. Hierarchtical variables
4. Multiple instances from one module definition.

Parameters

`parameter n=8;` defines a constant used at compile time. Change n to 16 to get a 16 bit shift register.

Multiple Instances

The first module `shftreg2` defines how a shift register is built but does not do it.

In the top module there are two instances of shift registers actually built.

Parameters Override[1]

Parameters can be overridden from a higher level using:-
module_name #(param1, param2, ...,paramN) instance_name(...
There must be N parameters defined in the module.
`shifty8` will be instantiated with its parameter still at 8.

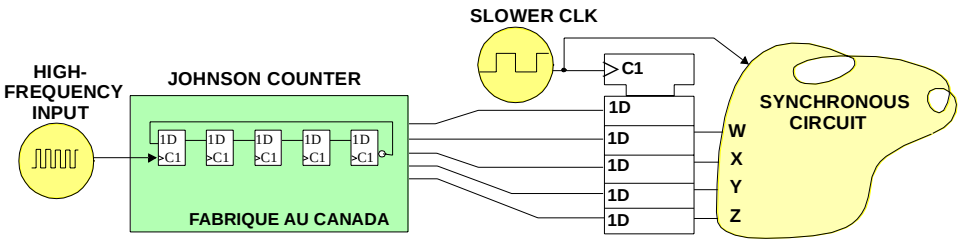
The Concatenate Operation

Note how one can concatenate the same register on both sides of an "=" or "<=" sign.

[1] Refering to parameter `n` outside the module `shifty1` by using `defparameter shifty16.n = 16` is an alternate method of parameter override used in simulation. It will not synthesize.
Another method, prefered for global parameters, is to use macros *`N parameter N=8;*

Application

- Use a fast Johnson counter to measure square-wave frequencies.
  Use the square-wave to clock the counter, and read VWXYZ with a slower circuit.
- The pattern tells exactly how many cycles have occurred between readings,
  provided it is less than ten.
- The Johnson counter is asynchronous to the main circuit.
  But the VWXYZ signals can be latched without races.
  There is no race if only one of the signal changes at a time.



This Johnson counter is a fast, asynchronously clocked, counter.
It is suitable for latching directly into a synchronous circuit.
However watch out for metastability

Mobius (Johnson) Counters

Mobius Strip

Note the right-hand flip-flop is inverted before it is fed back to the left-hand flip-flop.
Compare this to a paper Mobius strip. It is a paper ring glued together with a half twist. It became one ring when cut in half lenghtwise.

Speed

There are no gates, so this counter can be clocked as fast as the *toggle rate* of the flip-flops. This is several times faster than say a binary counter which has at an n-input AND or equivalent before the most significant flip-flop.

Size

This circuit becomes large an inefficent when large counts are needed. It has 2N states from N flip-flops.

One would use this as a prescaler for binary counter if large counts were needed.

▪ Shift Registers ▪