EXAMPLE 4.11 Develop a Verilog model for an interval timer that has clock, load and data input ports and a terminal-count output port. The timer must be able to count intervals of up to 1000 clock cycles.

SOLUTION The data input and counter need to be 10 bits wide, since that is the minimum number of bits needed to represent 1000. The module definition is

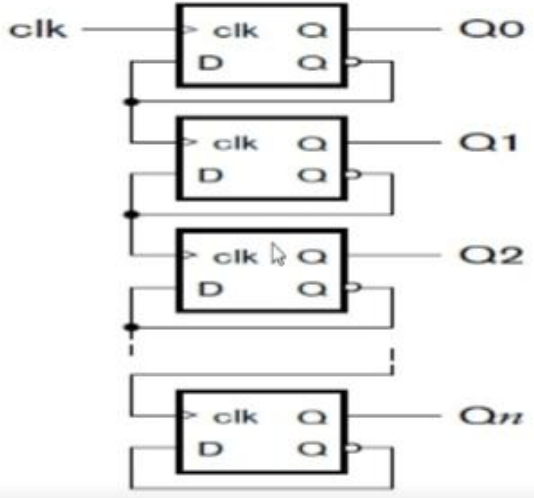
```
module interval_timer_rtl ( output       tc,
                            input [9:0] data,
                            input       load, clk );

  reg [9:0] count_value;

  always @(posedge clk)
    if (load) count_value <= data;
    else      count_value <= count_value - 1;

  assign tc = count_value == 0;

endmodule
```

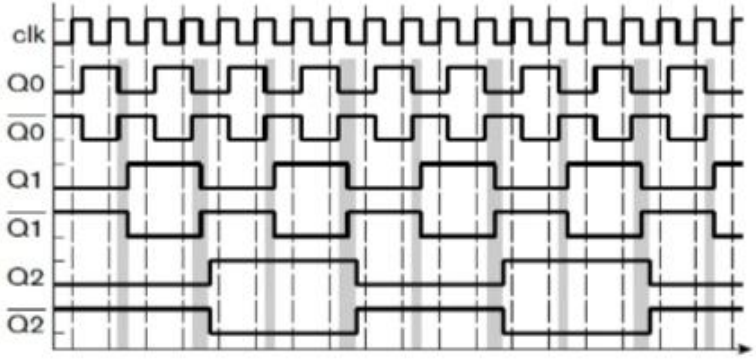
Modify the interval timer so that, when it reaches zero, it reloads the previously loaded value rather than wrapping around to the largest count value.

SOLUTION We need to use a separate register to store the data value to load into the counter. When the load input is activated, a new data value is loaded into the storage register as well as into the counter. When the terminal count is reached, the counter should be loaded from the storage register. The inputs and outputs of the revised interval timer are the same, so we don't need to change the ports of the module definition. The revised module is

```
module interval_timer_repetitive ( output       tc,
                                   input [9:0]  data,
                                   input        load, clk );

  reg [9:0] load_value, count_value;

  always @(posedge clk)
    if (load) begin
      load_value <= data;
      count_value <= data;
    end
    else if (count_value == 0)
      count_value <= load_value;
    else
      count_value <= count_value - 1;

  assign tc = count_value == 0;

endmodule
```

Ripple Counter



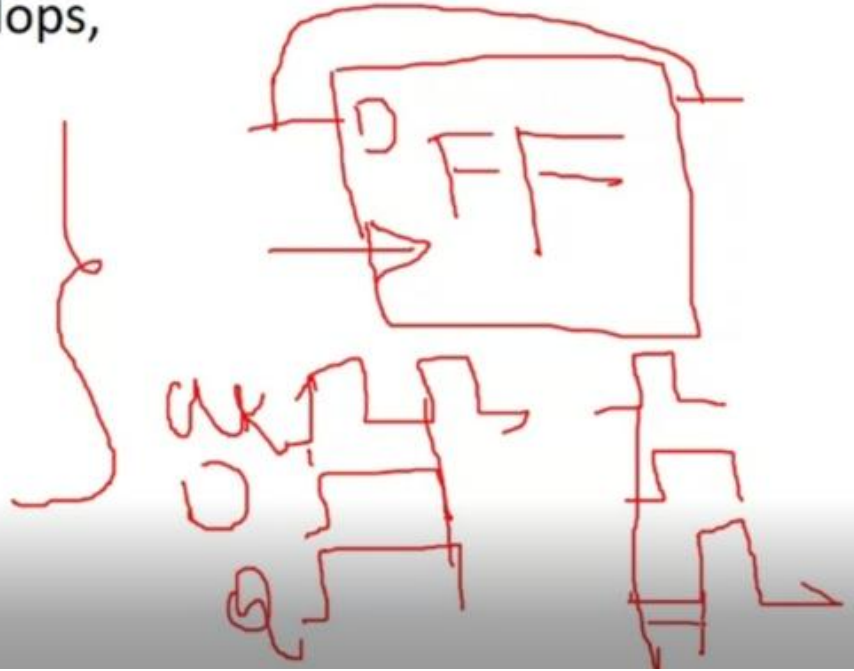


Propagation Delays, Setup, and Hold Times

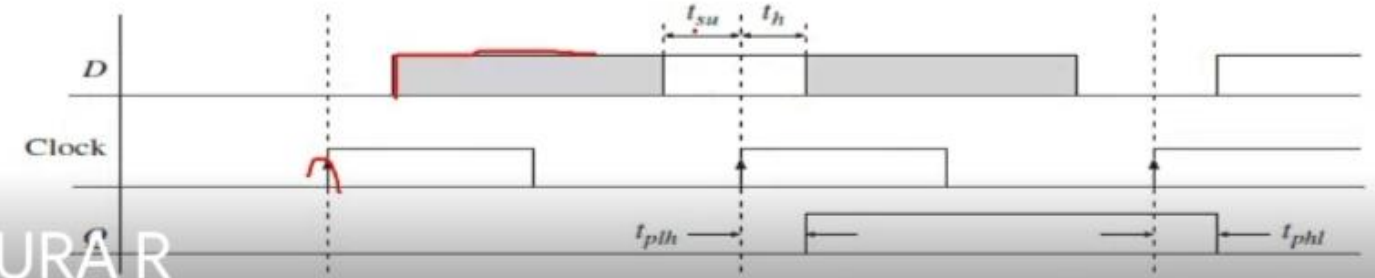
- There is a certain amount of time, albeit small, that elapses from the time the clock changes to the time the *Q* output changes. This time, called *propagation delay* or ***clock-to-Q delay*** of the flip-flop is indicated in Figure 1-34. The propagation delay can depend on whether the output is changing from high to low or vice versa. In the figure, the propagation delay for a low-to-high change in *Q* is denoted by t_{PLH}, and for a high-to-low change it is denoted by t_{PHL}.
- For an ideal *D* flip-flop, if the *D* input changed at exactly the same time as the active edge of the clock, the flip-flop would operate correctly. However, for a real flip- flop, the *D* input must be stable for a certain amount of time before the active edge of the clock. This interval is called the *setup time* (t_{su}). Furthermore, *D* must be stable for a certain amount of time after the active edge of the clock. This interval is called the *hold time* (t_h). Figure 1-34 illustrates setup and hold times for a *D* flip-flop that changes state on the rising edge of the clock. *D* can change at any time during the shaded region on the diagram, but it must be stable during the time interval t_{su} before the active edge and for t_h after the active edge. If *D* changes at any time during the forbidden interval, it cannot be determined whether the flip-flop will change state. Even worse, the flip- flop may malfunction and output a short pulse or even go into oscillation.

Sequential Circuit Timing

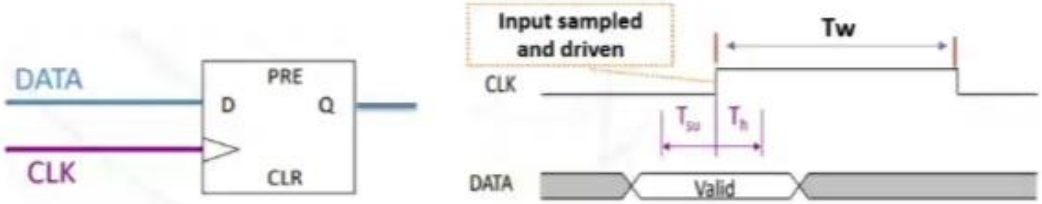
- The correct functioning of sequential circuits involves several timing issues. Propagation delays of flip-flops,
- gates and wires;
- setup times and
- hold times of flip-flops;
- clock synchronization;
- clock skew;



- Flip-flops typically have a setup time about 3–10x of the propagation delay of an inverter (NOT) gate. The hold times are typically 1–2x of the delay of an inverter. Minimum values for *tsu* and *th* and maximum values for *tplh* and *tphl* can be obtained from manufacturers' data sheets or ASIC (Application Specific Integrated Circuit) libraries accompanying design tools.
- su h

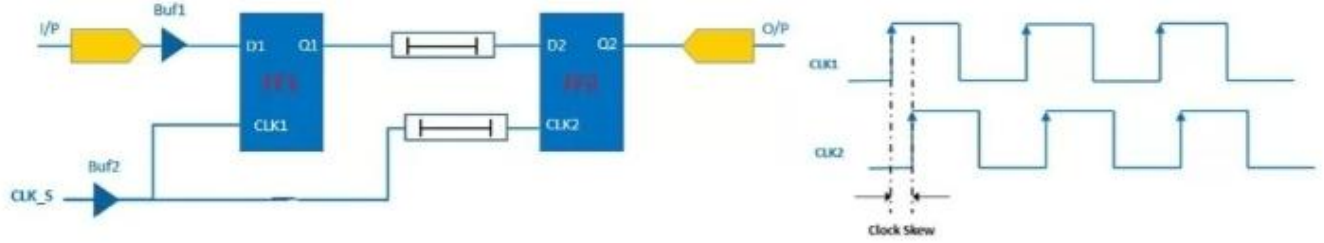


Set up and Hold Time



- In digital circuits, always data is either sampled or driven.
- Sampling is done with respect to reference clock and driving as well.
- **Pulse width (Tw)**
  - It is the time between the active and inactive states of the same signal
- **Setup time (T_{su})**
  - For an edge triggered sequential element, the setup time is the time interval before the active clock edge during which the data should remain unchanged
- **Hold time (T_h)**
  - Time interval after the active clock edge during which the data should remain unchanged

Clock skew



- Clock Skew is the difference in the arrival time of the clock at clock pin of different flip flops
- Correct behavior assumes that all storage elements sample at exactly the same time
- Not possible in real systems:
  - clock driven from some central location
  - different wire delay to different points in the circuit
- Problems arise if skew is of the same order as FF contamination delay
- Gets worse as systems get faster (wires don't improve as fast)
  - Distribute clock signals in general direction of data flow
  - Wire carrying the clock between two communicating components should be as short as possible
  - Try to make all wires from the clock generator be the same length - clock tree

Timing Conditions for Proper Operation

- In a synchronous sequential circuit, state changes occur immediately following the active edge of the clock.
- The maximum clock frequency for a sequential cir- cuit depends on several factors.
- The clock period must be long enough so that all flip-flop and register inputs will have time to stabilize before the next active edge of the clock.
- Propagation delays and setup and hold times create complications in sequential circuit timing.

- **Static timing analysis** (STA) is a method of validating the timing performance of a design by checking all possible paths for timing violations under worst-case conditions. A static analysis path starts at a source flip-flop (or at a primary input) and terminates at a destination flip-flop (or primary output). A static timing path between two flip-flops starts at the input to the source flip-flop and terminates at the input of the destination flip-flop. It does not go through the destination flip- flop. The path terminates when it encounters a clocked device. If a signal goes from register (flip-flop) A to register B and then to register C, the signal contains two paths. The timing paths in a synchronous digital system can be classified into 4 types:
    - Register to register paths (i.e., flip-flop to flip-flop)
    - Primary input to register paths (i.e., input to flip-flop)
    - Register to primary output paths (i.e., flip-flop to output)
    - Input to output paths (i.e., no flip-flop)