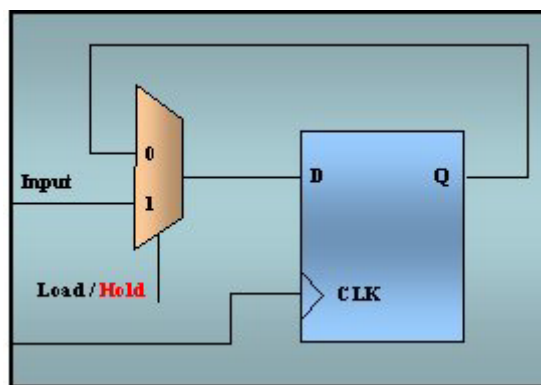


### Improving clock-gating for saving power

The gate count of ASICs is increasing drastically while factors such as area, cost & power are decreasing. Nowadays RTL coding is not restricted to front-end design as it used to be and designers are equally concerned with issues such as layout & power consumption.

Though Clock Gating is a well-known concept for power saving, for technologies below .18  $\mu\text{m}$  mere switching OFF of the clock is not good enough. For maximum power saving the data toggle also needs to be switched OFF with the clock. While tools such as Power Compiler are smart enough to know which signal is to be used for Clock Gating, they are still not intelligent enough to change the architecture of the RTL code.

Consider the following example, there is an input and depending on the logic it might be required to hold the signal at the output. This can be achieved by using a re-circulating flop to hold the value.



If the select line generates a *Hold*, the signal will only be asserted when it is required to hold the value (by default the input will always be passed to the flop).

OR

The same signal can be implemented as *Load*, which will be generated only when required (by default it will be 0).

If a gating element is added in the flop then the second implementation (*Load*) will save more power than the first implementation (*Hold*) since the data input to the flop will only be toggled when required.

#### Example of incorrect code

```
always @(posedge clk)
  if (ack == 1'b0)
    out <= out; // Hold the value if no ack
  Else
    out <= in ; // Default - always pass the value irrespective of valid command
```

#### Example of better code

```
always @(posedge clk)
  If (ack == 1'b0)
    out <= out ; // Hold the value if no ack
  Else if (valid_req)
    out <= in ; // Load only if there is a valid request
  Else
    out <= IDLE_cmd; // Default - just keep IDLE command
```