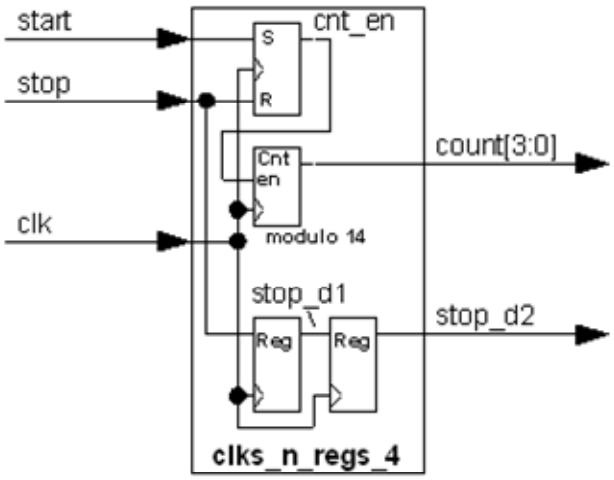
Verilog by Example, errata

We now introduce a few common state-type operations to show how increasingly sophisticated register-based functions are implemented in always-blocks. A four-bit counter is enabled by a "start" event, and stopped by a "stop" event. The SR flop allows the start and stop events to be short, e.g. one-clock pulses, rather than a continuously enabling flag. Additionally, for further illustration, we delay the ~~start~~ stop signal two clocks and send it out.

You'll notice that we have not shown the asynchronous reset. This is done ~~for clarity; from this~~ point forward it is assumed. It is implemented in the code, and always will be (in this book).



SR flop and counter

```
////////////////////////////////////////////
// SR flop and counter
////////////////////////////////////////////

module srflop_n_cntr (  clk,
                        reset,
                        start,
                        stop,
                        count
                      );

    input           clk;
    input           reset;
    input           start;
    input           stop;
    output [3:0]    count;
    output          stop_d2
    reg             cnt_en;
    reg [3:0]       count;
    reg             stop_d1;
    reg             stop_d2;

    // ------ Design implementation -----

    // SR flop
    always @( posedge clk or posedge reset )
      begin
        if ( reset )
          cnt_en <= 1'b0;
        else if ( start )
          cnt_en <= 1'b1;
        else if ( stop )
          cnt_en <= 1'b0;
      end
```