Multi-rate Hardware Clocking

- Example to imitate a clock frequency of \( \text{freq/4} \)

```verilog
reg [1:0] count, count_c; // two bits counts 00, 01, 10, 11, 00, ...
always @(*) begin
    // default. However no need to wrap counter back to 00 with four counts
    count_c = count_c + 2'b01; // just let the counter wrap 2'b11 \( \rightarrow \) 2'b00
end

wire en_freq4; // use a wire in this example
assign en_freq4 = (count == 2'b00) // en_freq4 will be high 1/4 of the time.
    // We could have chosen any value of count.

always @(posedge clk) begin
    count <= #1 count_c;
    if (en_freq4 == 1'b1) begin
        Q <= #1 D;
    end
end
```

PLL

\( \text{en_freq4} = 1 \)

Every fourth cycle is equivalent to using \( \text{freq/4} \)
Multi-rate Hardware Clocking

- Example to imitate a clock toggling at 1 Hz, with 500 MHz clock

```verilog
reg [28:0] count, count_c; // 29 bits counts up to 536 million
reg en_increment; // use a reg in this example
always @(*) begin
    // defaults
    count_c = count_c + 29'h0000_0001; // a flip-flop register
    en_increment = 1'b0; // a combinational logic signal

    if (count == 29'd499_999_999) begin
        count_c = 29'h0000_0000; // wrap counter back to zero
        en_increment = 1'b1; // pulse FF enable signal high
    end
end

always @(posedge clk) begin
    count <= #1 count_c;
    if (en_increment == 1'b1) begin
        Q <= #1 D;
    end
end
```