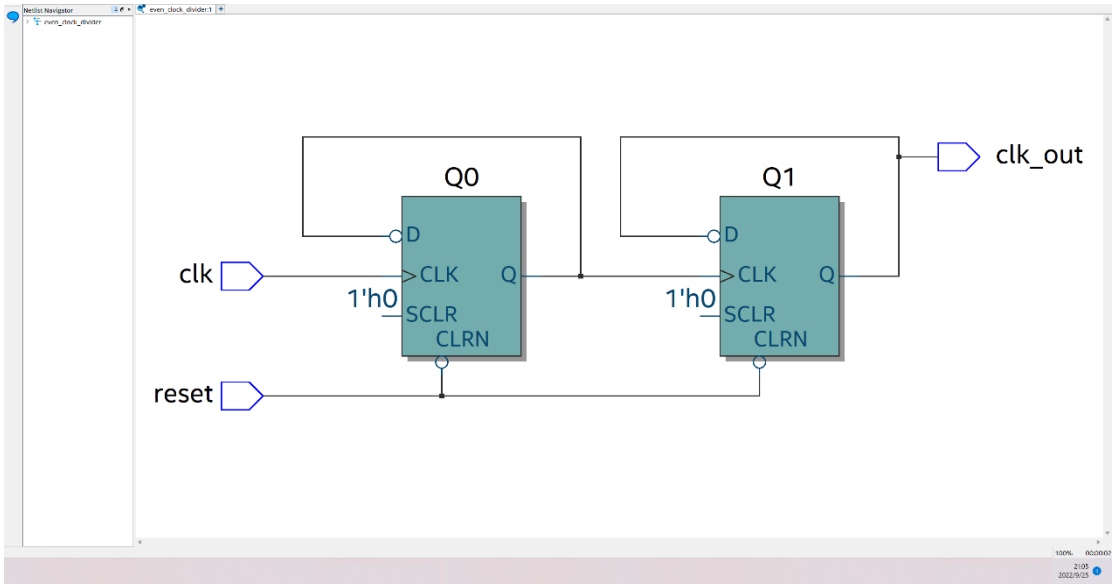
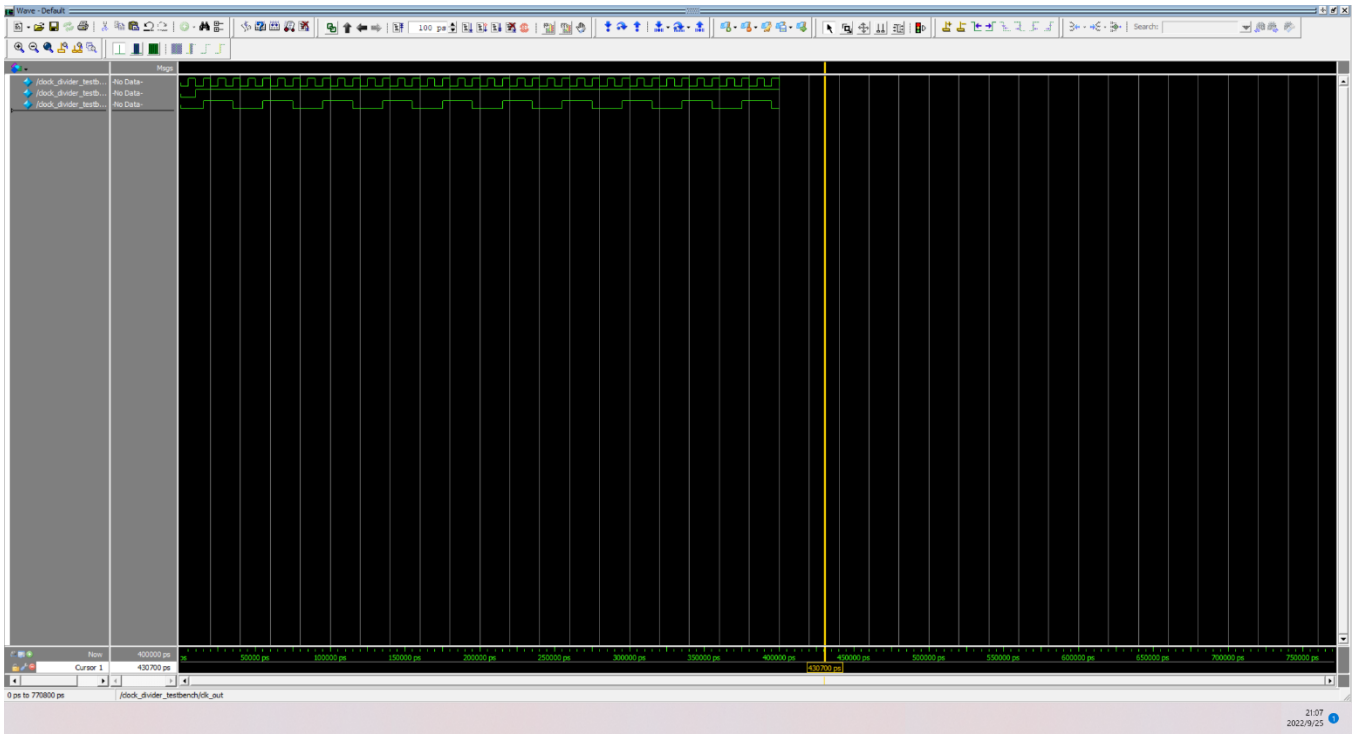
Even clock divider

The clock is divided by 4, done by 2 D flip flops, while the first output was its inverse input. The second was the same. The second ff's clock is the first ff's output, so each ff can divide the clock frequency by 2, thus 2 ff that connects together is divided by 4.

The reset is to control whether the ffs are on or off.



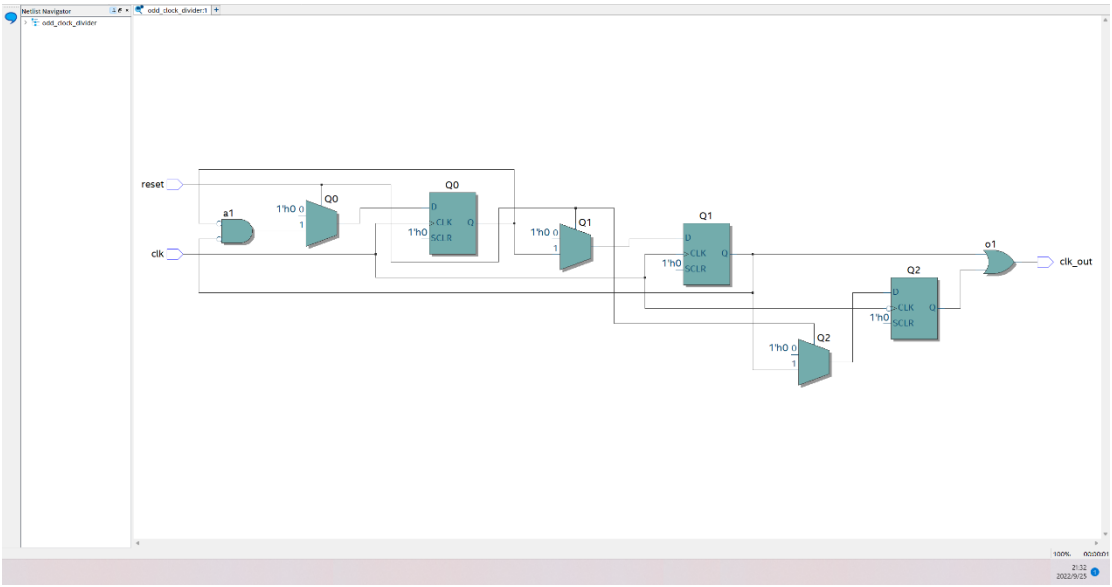
As the waveform shows, when both ff turns on, the output frequency is 4 times than the original clock.



Odd clock divider

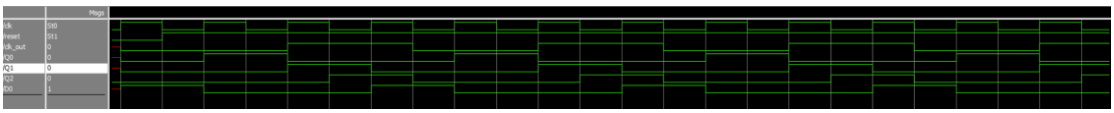
The clock is divided by 3, done by 3 D flip flops, while the first output was the second input. The second output was third input, and the output clock frequency was the second output OR third output. The first ff's input is the inverse first output AND inverse second output. All the ff's clock are the origin clock, but the third clock lies on negative edge instead of positive edge.

The reset is to control whether the ffs are on or off.

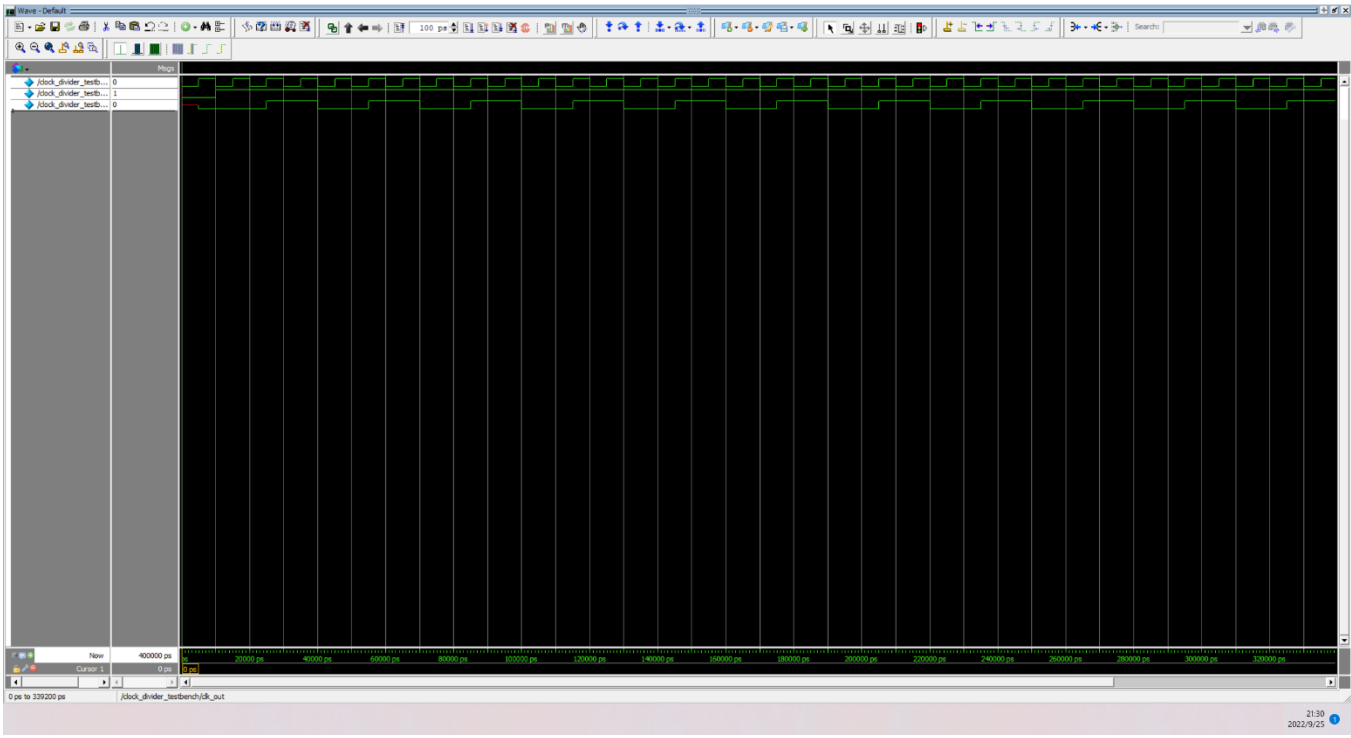


For Q0's input, because of the ff only reads the tick before, so when Q0 and Q1 are both low, they will last 2 ticks(D0). In the first tick, the clock is 0, so Q0 would not rise, so Q0 would rise 2 ticks after Q1 is changes from high to low(1 tick for reading previous input, 1 tick for wait clock to rise to high).

For the output, because Q2 reads the negative of the clock, so it would be 1 tick slow than Q1, thus Q1 OR Q2 would provide a 3 tick high output.

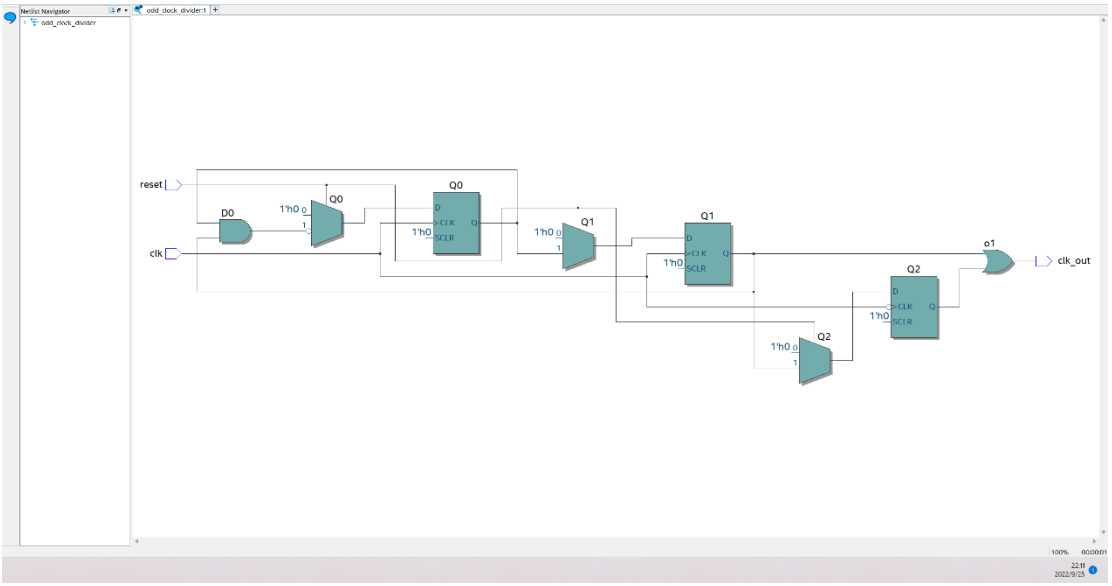


As the waveform shows, when both ff turns on, the output frequency is 3 times than the original clock.

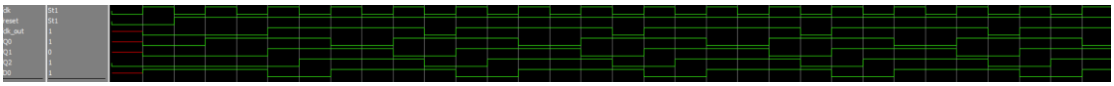


Bias clock divider

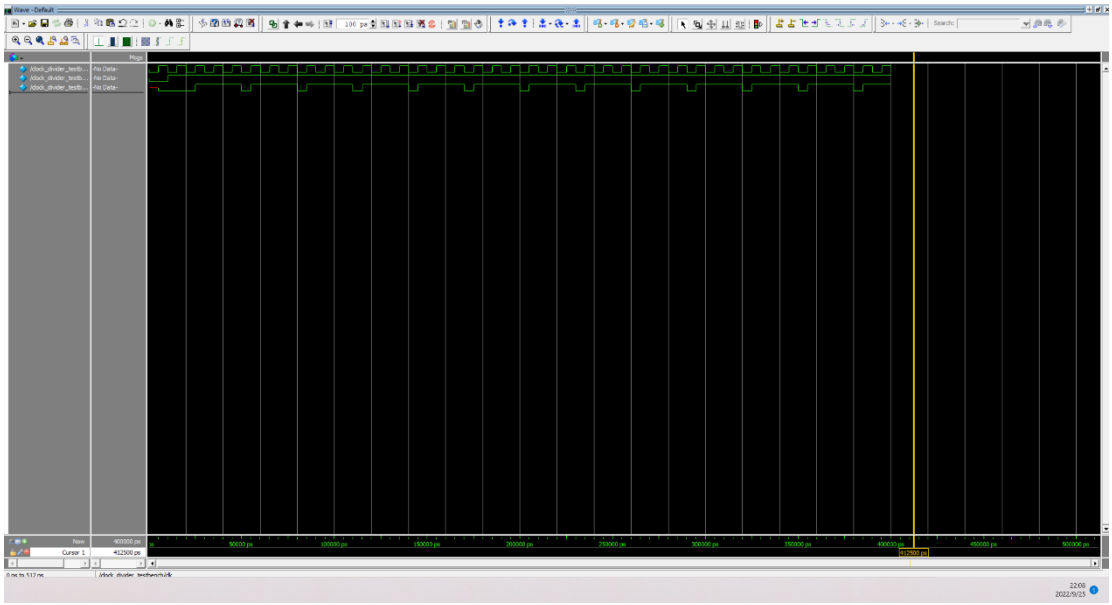
It is basically same as the divider before, but the input of the first ff is the inverse first output AND inverse second output.



This helps to generate 2 tick low, but not 2 tick high, so all the ffs will have 4 tick high and 2 tick low. Thus, Q1 OR Q2 is 4 tick high and 2 tick low, which is biased.



As the waveform shows, when both ff turns on, the output frequency is 3 times than the original clock, with 67 duty cycle.



Testbench

```
module clock_divider_testbench();

logic clk, reset, clk_out;

timeunit 1ns;
timeprecision 100ps;

odd_clock_divider dut(clk, reset, clk_out);

initial
	begin
		reset = 0;
		clk = 0;
		  #10 reset = 1; //turn on flip flop
	end

always
	begin
		#5 clk = 1;
		#5 clk = 0;
	end

endmodule
```