# Experiment 1 – Clock and Periodic Signal Generation

Borna
Tavassoli,
810198374

*Keywords*— **Clock, Ring Oscillator, LM555, Schmitt Trigger Oscillator, Frequency Divider, 74HCT, MAXPLUS2, Modelsim, Quartus, LTspice**

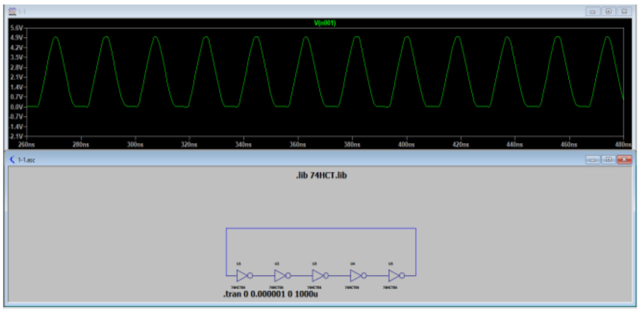
## I. Introduction

Today, a huge number of circuits depend on their clock inputs and frequency to function properly. In this experiment we deal with 3 different kinds of such clock generators.

Ring Oscillator, LM555 IC (in astable mode) and Schmitt Trigger Oscillator.

## II. Clock generation using ICs and analog components

### A. Ring Oscillator



Above is a simulation of a ring oscillator in LTspice using 5 inverters (74HCT04).

T = 2N * delay(inv)

And from the picture, we estimate T to be near 20.

So, 20 = 2 * 5 * delay(inv) therefore, delay(inv) is roughly 2ns.

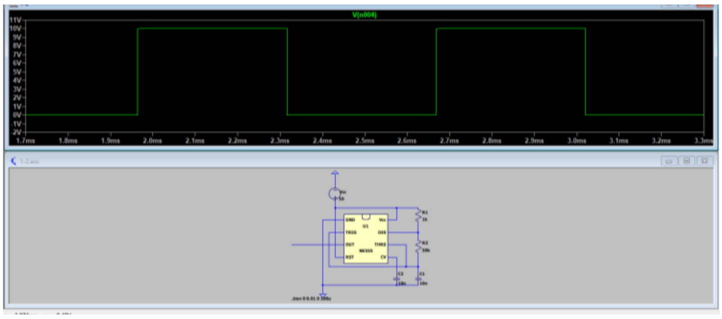
### B. LM555 timer

We can also use an LM555 IC to produce a clock signal. The duty cycle can be modified by changing a resistor value in the circuit. First the timer is tested in astable mode and then 3 different values for R2 are tested and the frequency and duty cycle for each one is calculated.

Note that I have used NE555 IC instead of LM555 since it was not available in my current version of LTspice .

Note that the capacitors are all 10nF and R1 = 1 KΩ

1. For the first part we have simply implemented LM555 with the following values
   R1 = 1KΩ, R2 = 50KΩ
   T = 0.693 × (1 + 100) × 10^3 × 10 × 10^(-9) = 0.69ms
   f = 1 / T = 1.44GHz
   Duty Cycle = 50.49%



2.1.
R1 = 1KΩ, R2 = 1KΩ
T = 0.693 × (1 + 2) × 10^3 × 10 × 10^ (-9) = 0.02ms
f = 1 / T = 0.05MHz
Duty Cycle = 66.66%

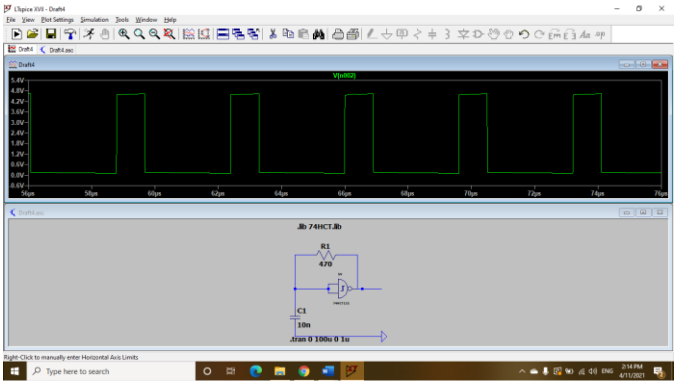
2.2.
R1 = 1KΩ, R2 = 10KΩ
T = 0.693 × (1 + 20) × 10^3 × 10 × 10^ (-9) = 0.14ms
f = 1 / T = 0.07MHz
Duty Cycle = 52.49%

2.3.
R1 = 1KΩ, R2 = 100KΩ
T = 0.693 × (1 + 200) × 10^3 × 10 × 10^ (-9) = 1.39ms
f = 1 / T = 0.007MHz
Duty Cycle = 50.2%

### C. Schmitt trigger oscillator

Note that we have used a 74HCT14 in our circuit.

f = α / RC ➔ α = RC / T

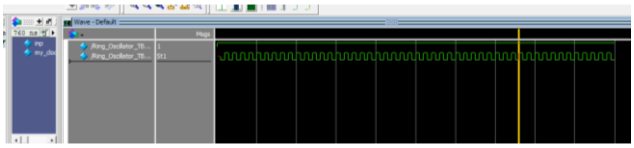


1. R = 470Ω, dx = 0.004ms ➔ α = 1.175
2. R = 1Ω, dx = 0.007ms ➔ α = 1.42
3. R = 2.2Ω, dx = 0.015ms ➔ α = 1.46

## III. FPGA Design

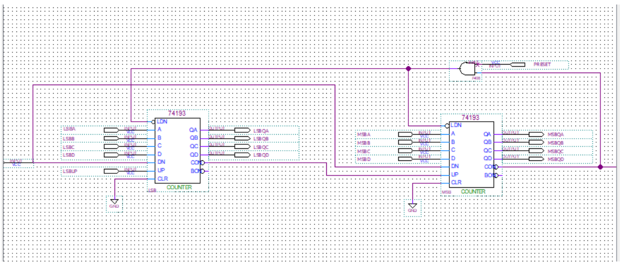
### A. Ring Oscillator

We redo the ring oscillator part now in Verilog. If we run the testbench the result is as follows:

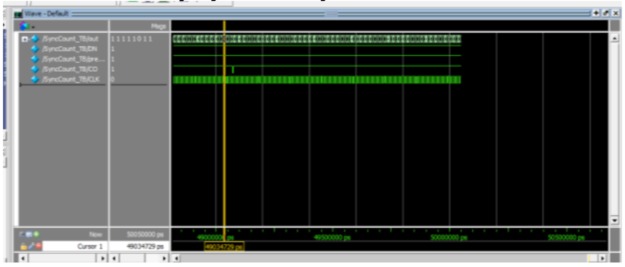


As we had expected, T = 20ns. So, the frequency is 1/T = 5*10^7Hz.

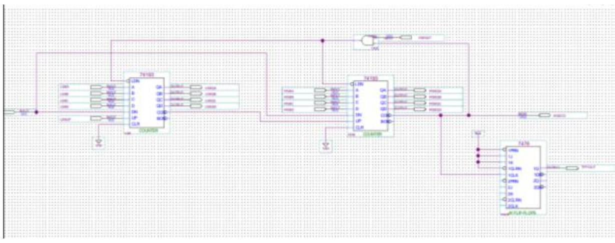
## *B. Synchronous Counter as a Frequency Divider*



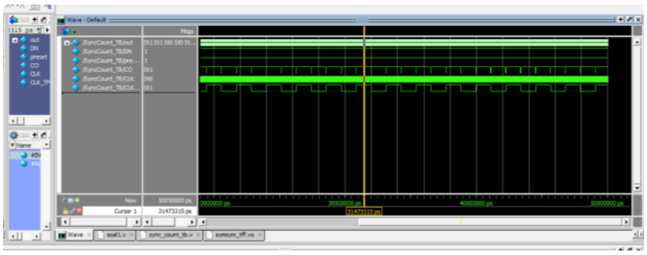
As shown in the picture above, we have used two cascaded blocks of 74193 IC with a pre-setting mechanism. When the 4-bit LSB counters reaches 1111, the MSB is incremented by 1. Parallel loading is used to start the counting with 10001110; So, it takes 255 - 142 (10001110) = 113 clock cycles for out to iterate a full cycle and then its value becomes 10001110. As you can see the duty cycle is very little!



## *C. T Flip Flop*



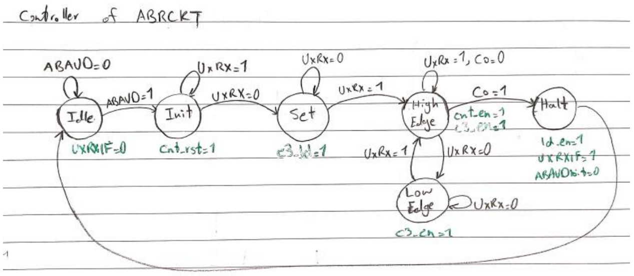
Now that we've used a TFF, the duty cycle will be 50% as shown below:



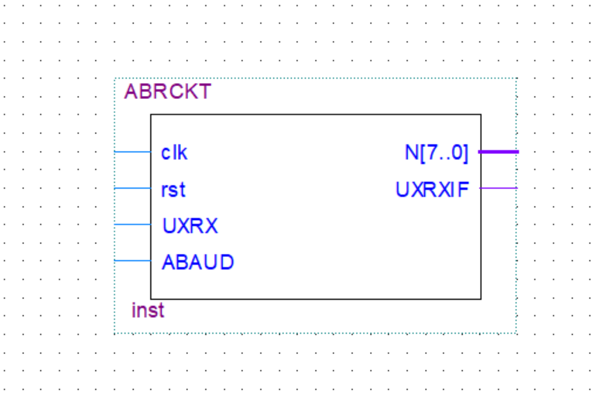
## IV. BAUD RATE GENERATOR FOR UART SERIAL COMMUNICATION

## *A. Automatic Baud Rate Calculator*

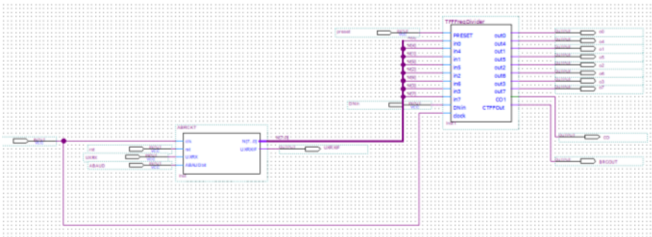
First, we need to draw the controller:



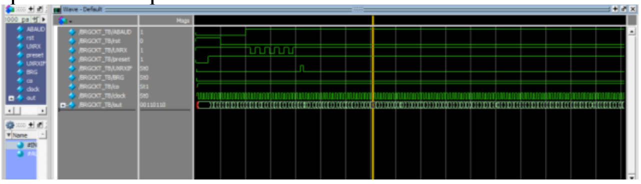
Then we implement the controller along with its data path to generate ABRCKT:



Now using the said component, we synthesis BRGCKT as shown below:



If we test the newly built circuit, we notice that the frequencies are quite similar!



Note that:

t1 = 3118180, t2 = 9046180 ➔ dt = 5928000ps

➔$F_{BRGCKT}$ = 168,690

T = 2 * 2 * 3 = 12 * $10^{-9}$, N = 28 ➔ $F_T$ = $10^9$ / (12 * 2 * 228) = 182,748 which has about 8 percent error.

## V. Conclusion

Having done these experiments, we have learned to use not only regular clock, but also, custom made clocks for different purposes. We were able to build a Ring Oscillator and use it to build Frequency Dividers, and later on, Baud Rate generators. We also got familiarized with UART and its different implementations.