Course Name: Digital Hardware Design
Course Code: 17B1NEC741

Clock Dividers

Dr. Arti Noor
Dean, Academic Affairs
Electronics and Communication Engineering,
Jaypee Institute of Information Technology, Noida

Introduction

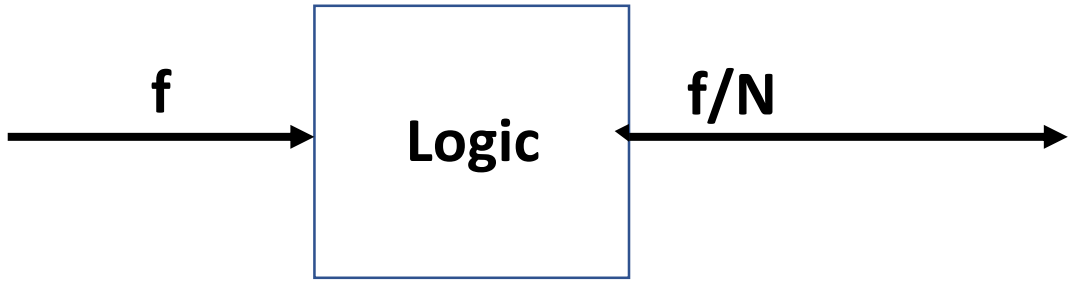
- Sequential circuits need a reliable clock signal for circuit operations.
- Digital systems need several different clock signals to drive different subsystems.
- Cock divider circuit creates different frequency clock signals from an input clock source.

Clock Frequency Dividers for f/n in three cases:

- Where n is an odd integer number
- Where n is even integer number
- Where n is a decimal number

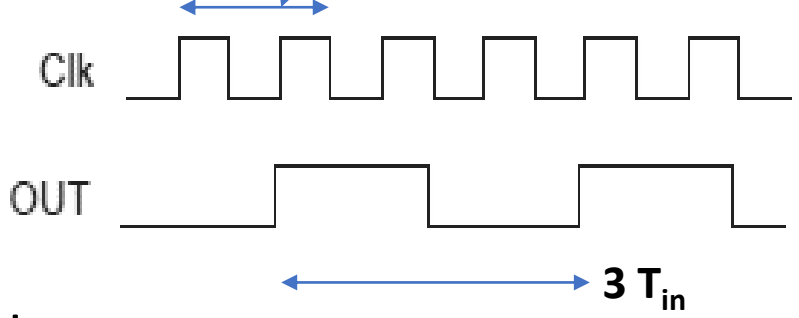
Frequency Divider for f/n :odd number

Example n=3



For N=3

$T_{out}=3\ T_{in}$



$T_{out}=N\ T_{in}$

$F=1/T$

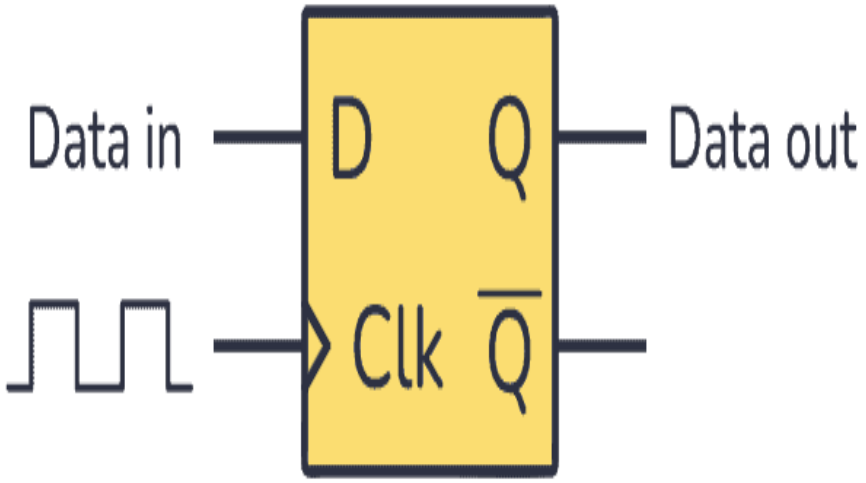
Requirement:

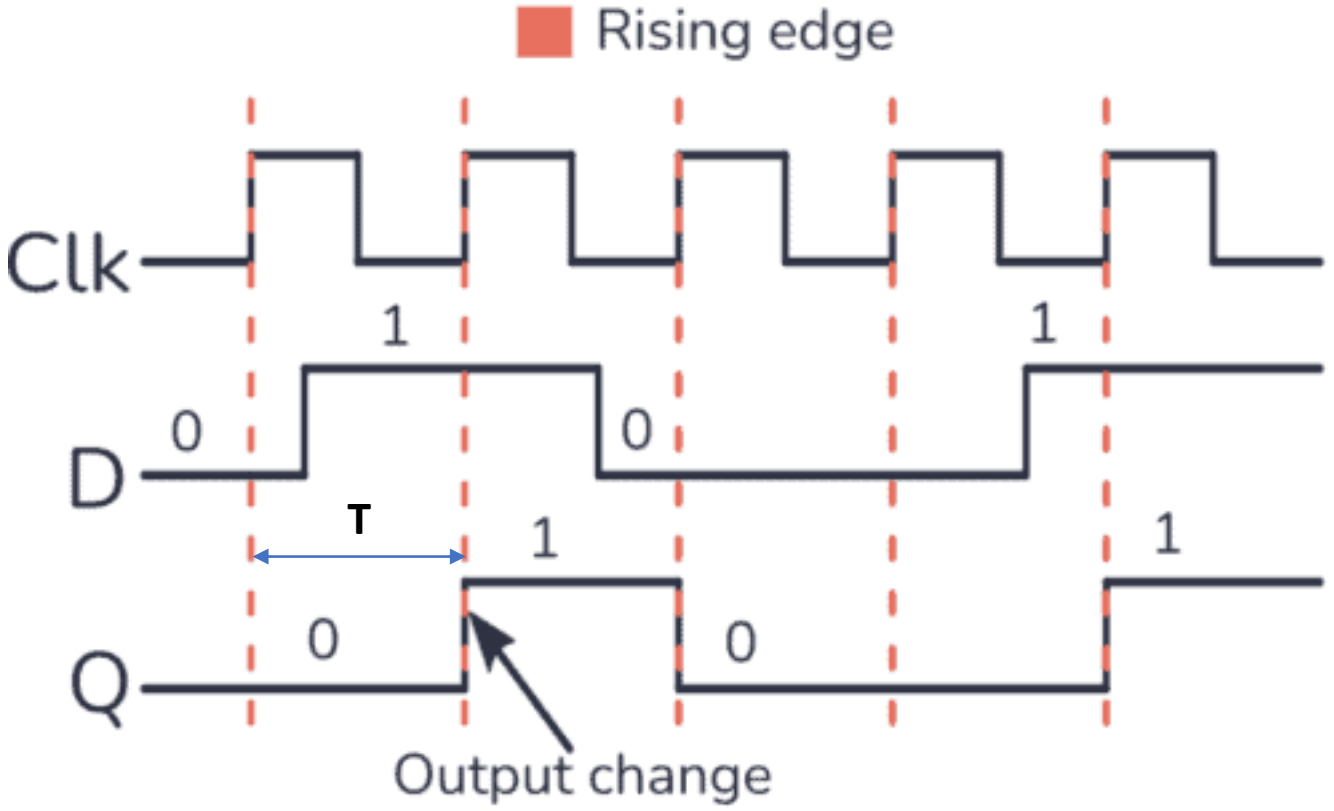
1. Mod N counter
2. DFFs
3. And some Logic

Frequency Divider for f/n :odd number

Example n=3

DFF





Q is delayed by T for +clk edge

Q is delayed by T /2 for -clk edge

T

Mod 3 counter

- It can be constructed by using 2 FF(2^2) because FF required for mod 3 counter.
- The no. of states required for mod counter is three states 00, 01, 10 and the final state is xx.

MSB	LSB	Decimal value
0	0	0
0	1	1
1	0	2
x	x	x

Clock divide by 3

- A divide by 3 clock requires a mod 3 counter.
- It can be constructed by using 2 FF(2^2) because FF required for mod 3 counter.
- The no. of states required for mod counter is three states 00, 01, 10 and the final state is xx.
- The output of the clock divide by three is not 50% duty cycle. The duty cycle will be 75% if the output is 1,1,0

Current state		Next state		Output
Qb	Qa	Qb+	Qa+	
0	0	0	1	1
0	1	1	0	1
1	0	0	0	0
x	x	x	x	x

Clock divide by 3

The K-Map realization for input FFA

Qa \ Qb	0	1
0	1	0
1	0	x

Da=Qa'Qb'

The K-Map realization for input FFB

Qa \ Qb	0	1
0	0	0
1	1	x

Db=Qa

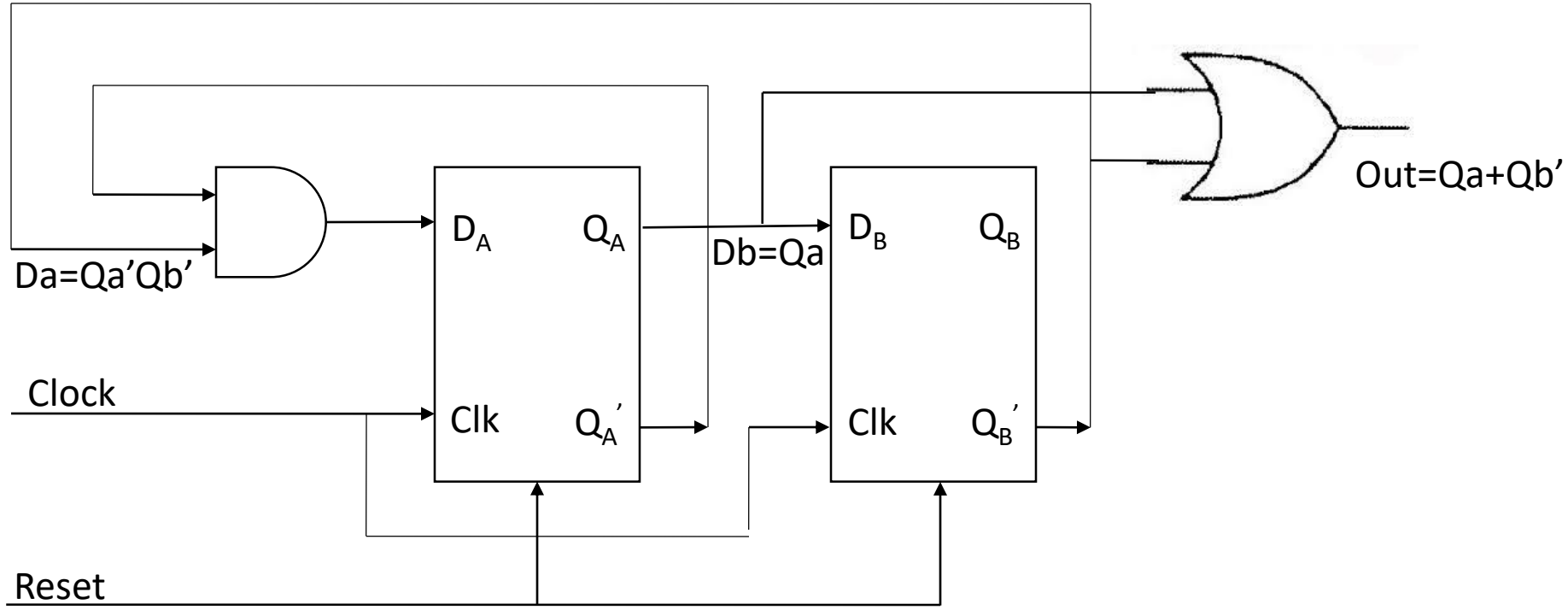
The K-Map realization for output of the FFB

Qa \ Qb	0	1
0	1	0
1	1	x

Out=Qa+Qb'

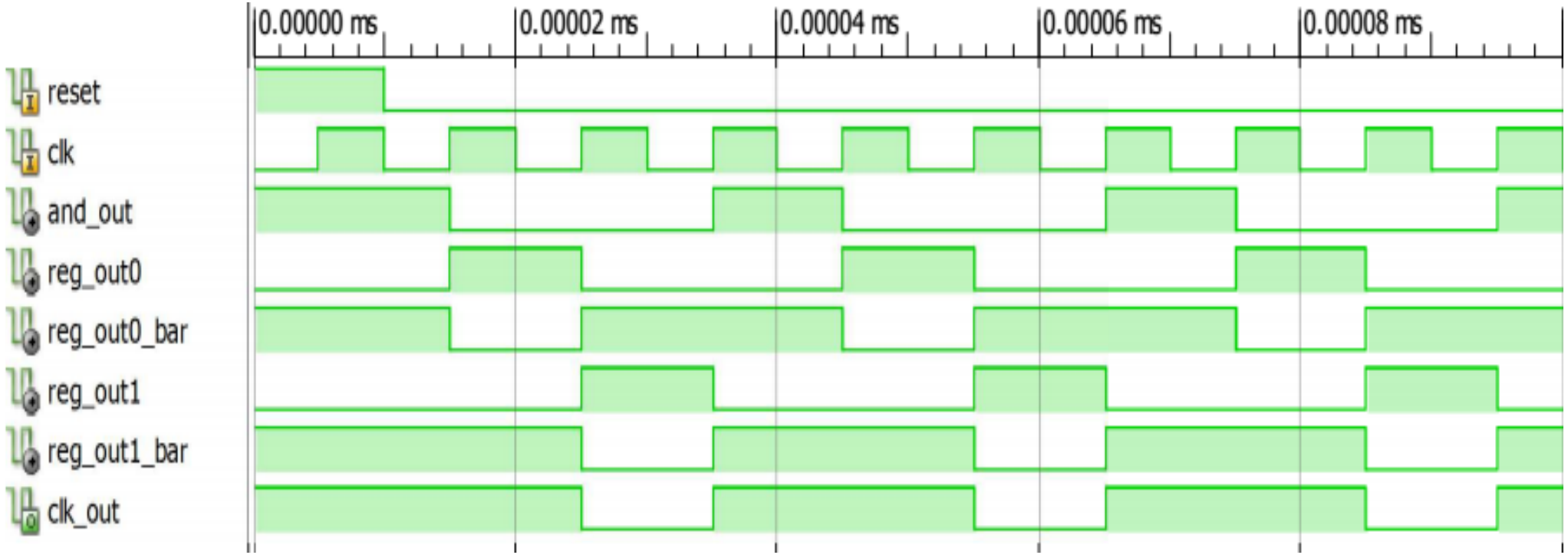
Clock divide by 3

- The micro architecture of the clock divide by 3 is



Clock divide by 3

- **Timing diagram for clock divided by 3 without 50% duty cycle**

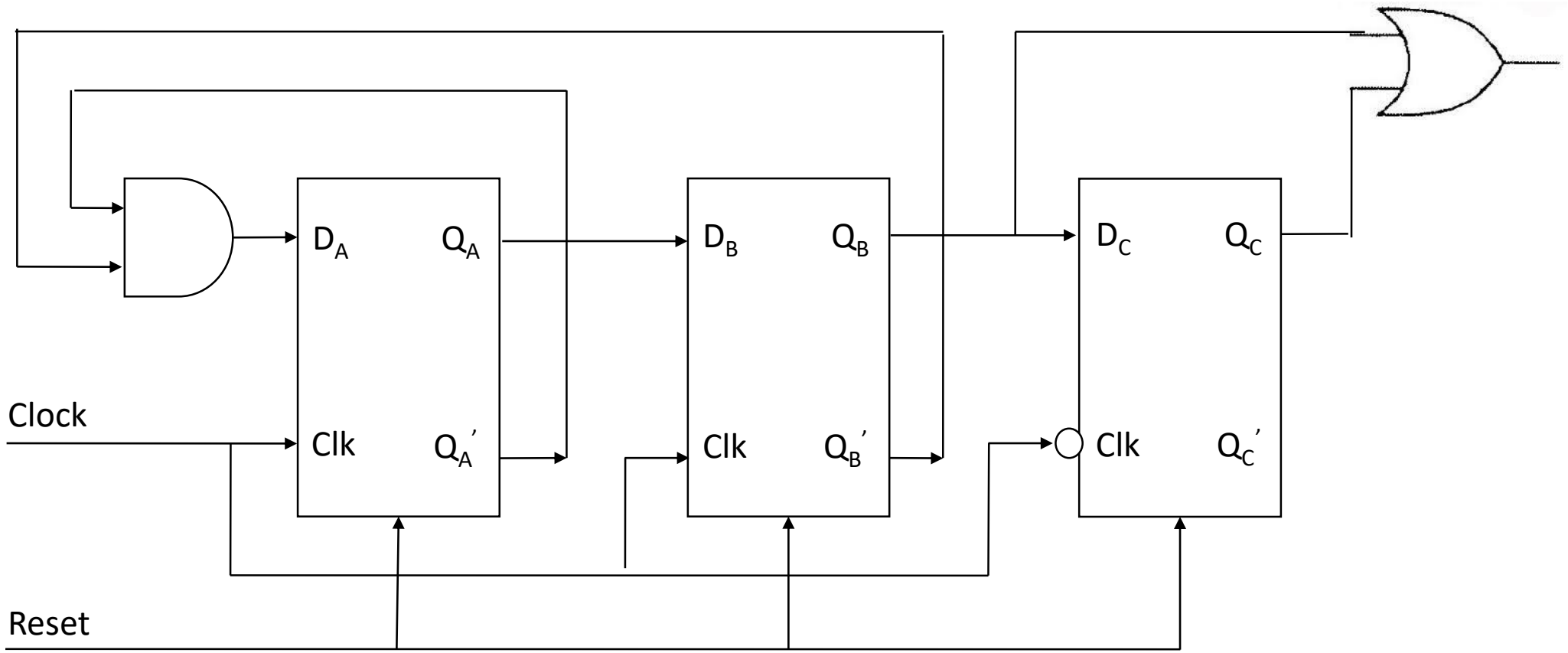


Input : Clock and reset

Output : clk_out

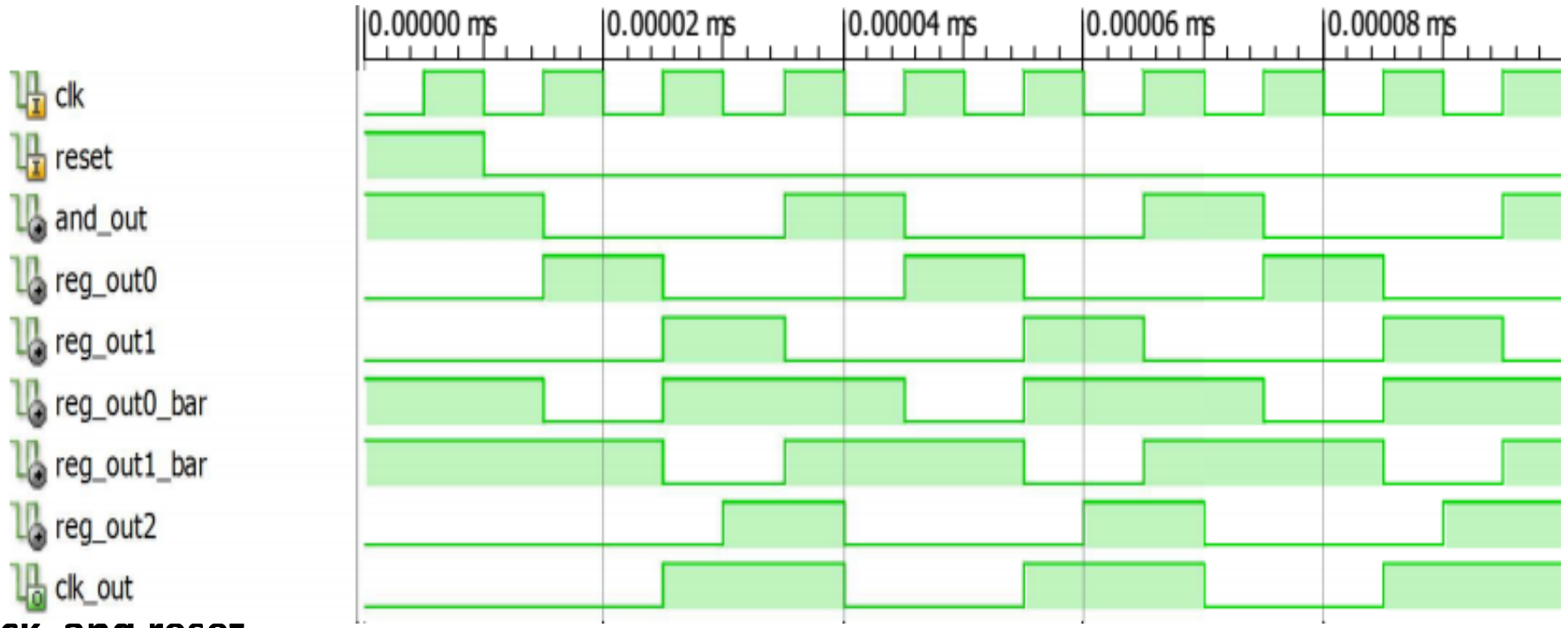
Clock divide by 3

- To get 50% duty cycle the out of the FFB will as input to the negative edge trigger FF.
- The output of the third FF and the output of the second FF is given as input to the OR gate.



Clock divided by 3

Timing diagram for clock divided by 3 with 50% duty cycle.

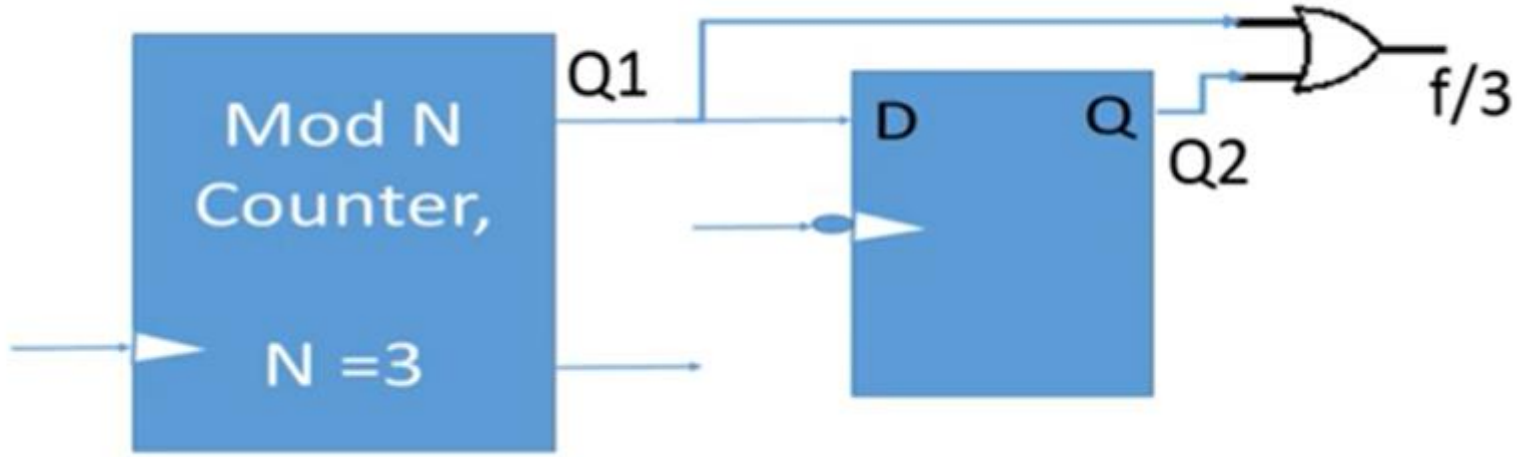


Input : Clock and reset

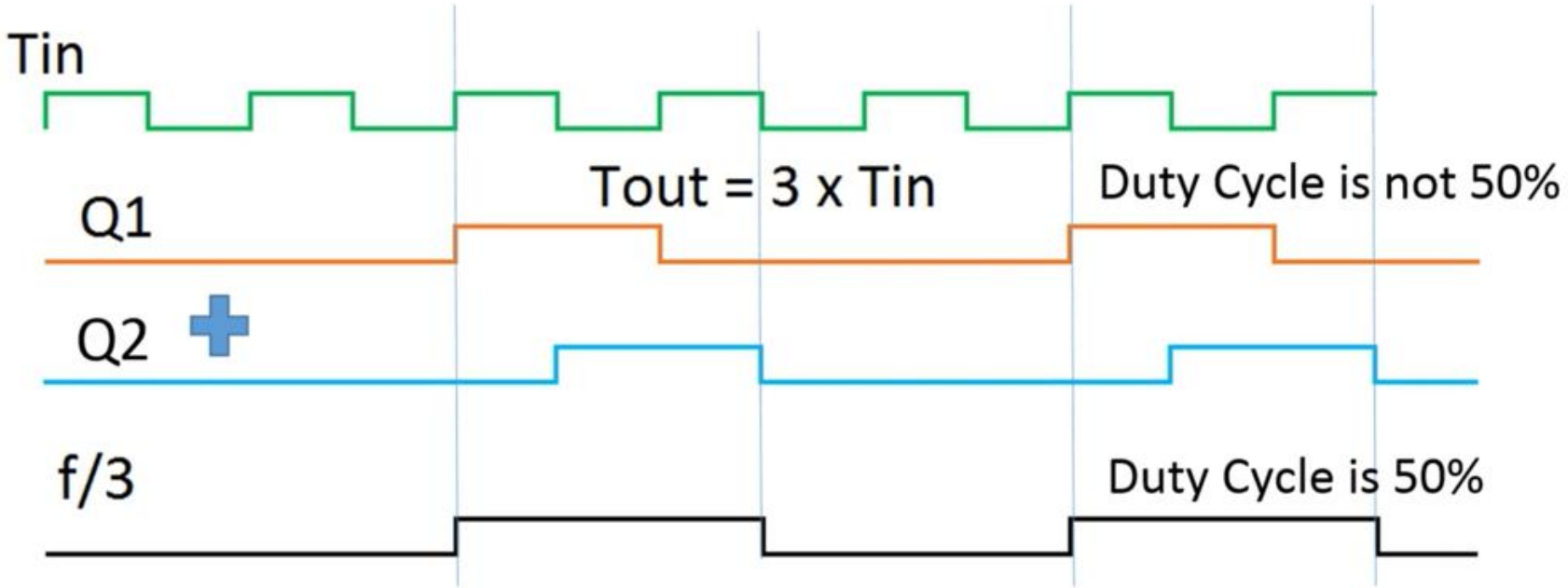
Output : clk_out

Generic Circuit for Clock divided by 3

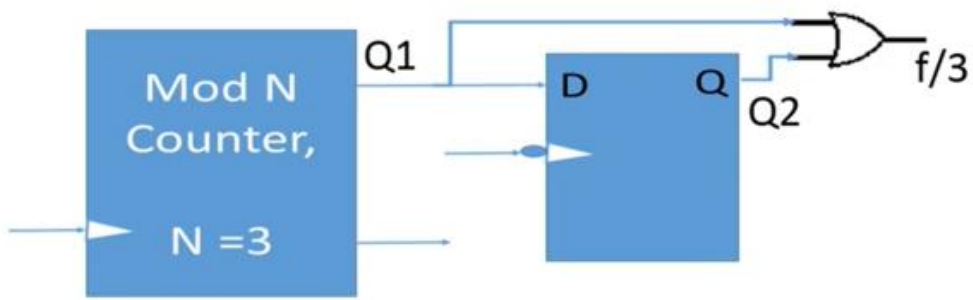
Q1	Q0
0	0
0	1
1	0



Frequency Divider for f/3



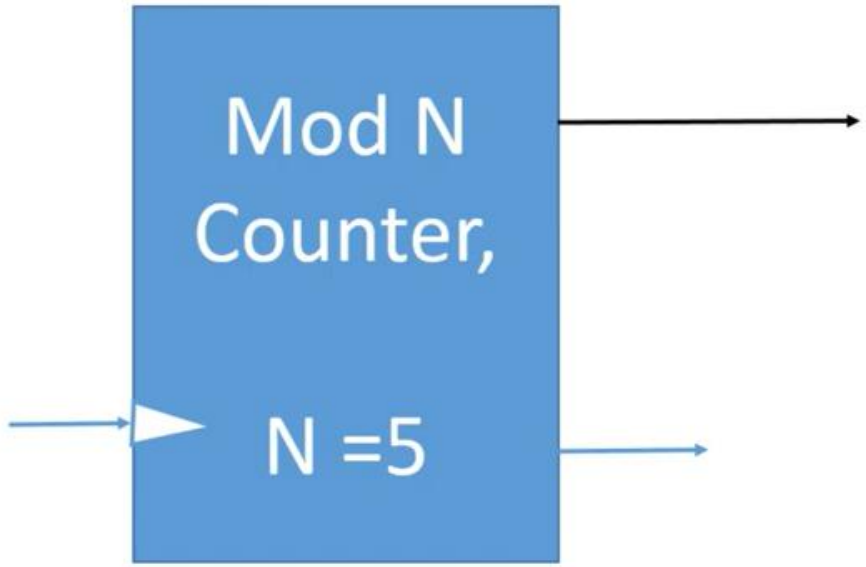
Q1	Q0
0	0
0	1
1	0



Clock divider by 5

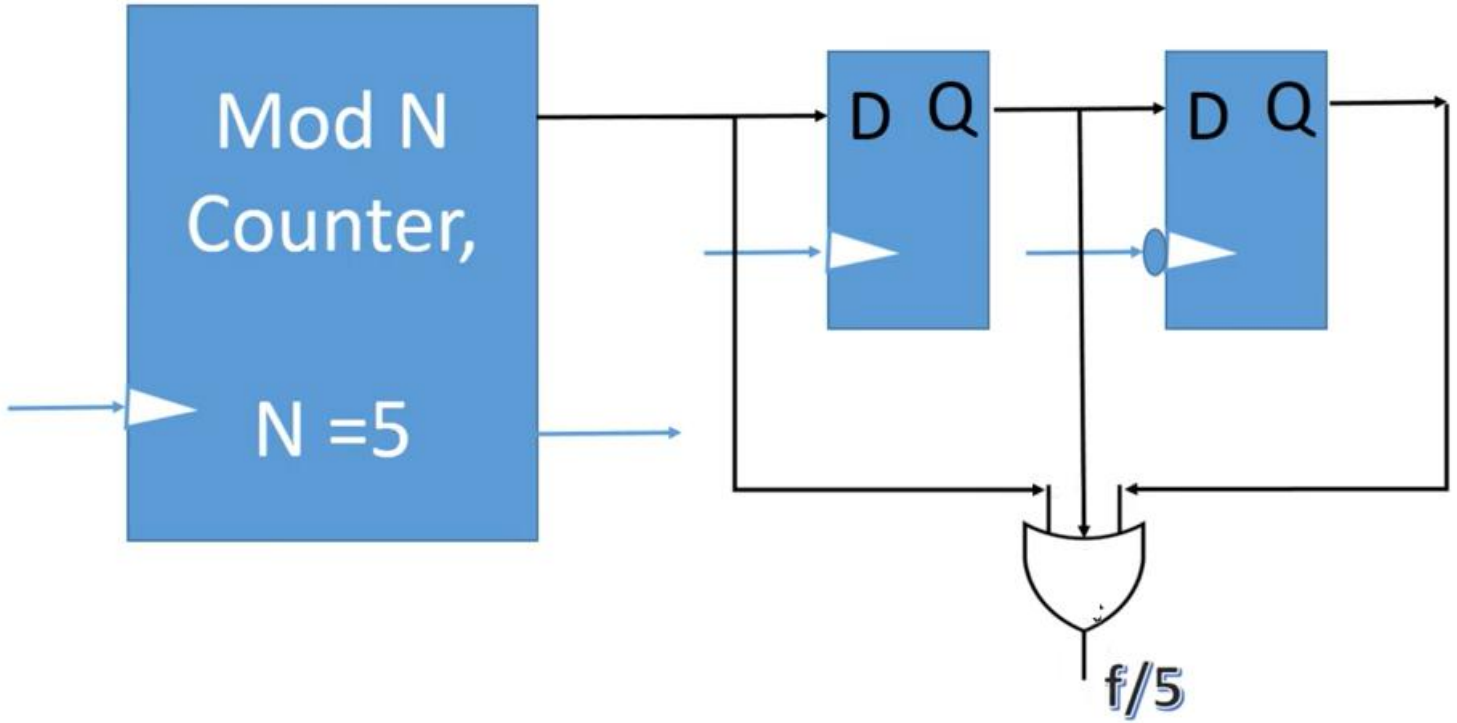
Frequency Divider for f/5 and Duty Cycle 50%

Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0



Clock divider by 5

Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0

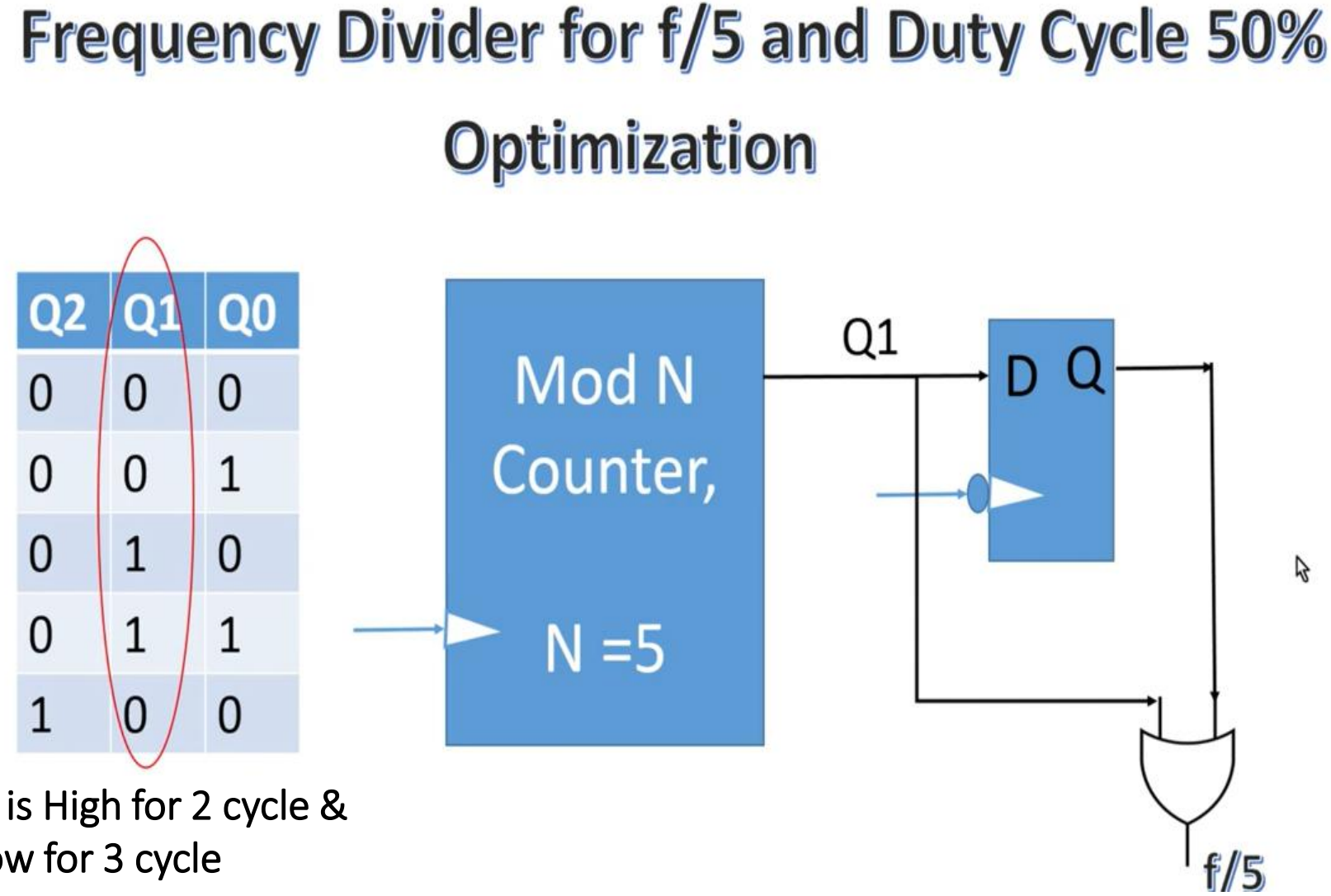


Clock divider by 5

Alternative Method

Pass the output of the second FF to one more FF which is triggered with negedge of clk

then make ORing of these two.



It is High for 2 cycle & low for 3 cycle

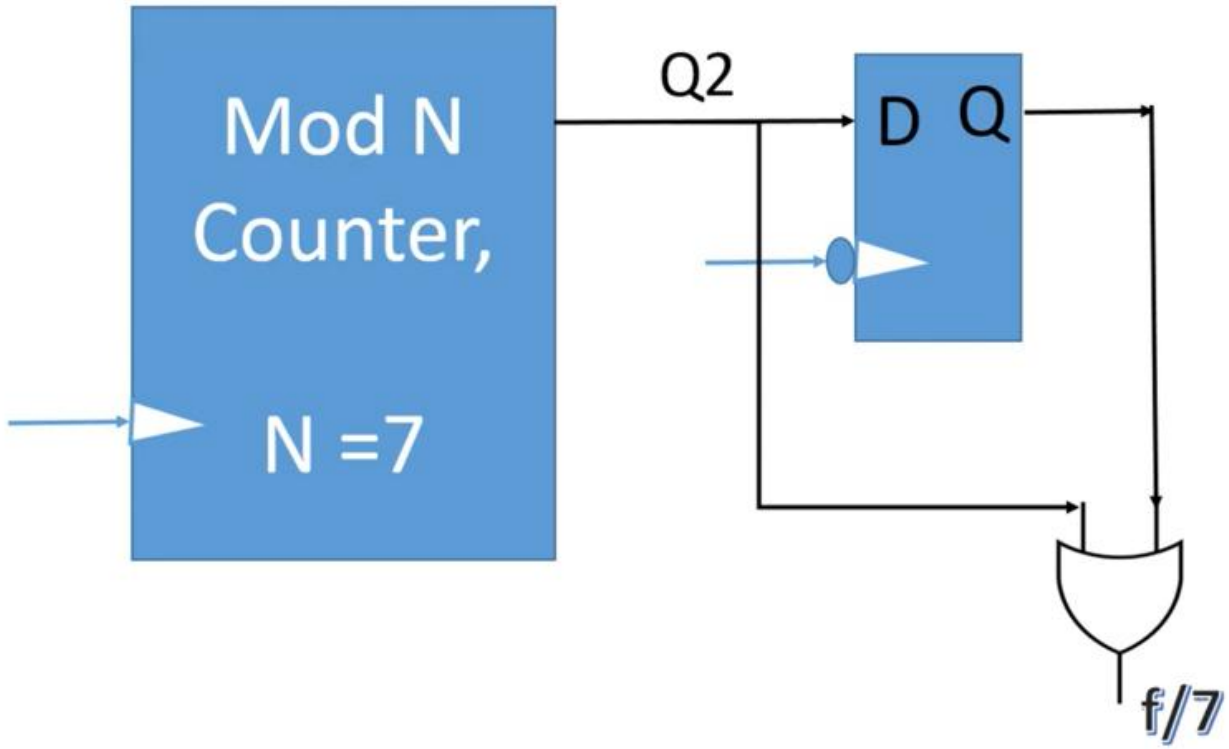
Conclusion for implementation of f/n divider

- Need to include ModN Counter
- Positive edge triggered flip flop alters output by 1 clock cycle.
- Negative edge trigger flip flop alters output by ½ clock cycle.
- Need to choose one of the output of ModN counter having frequency f/n and less alteration required for hardware optimization.

Frequency Divider for f/7 and Duty Cycle 50%

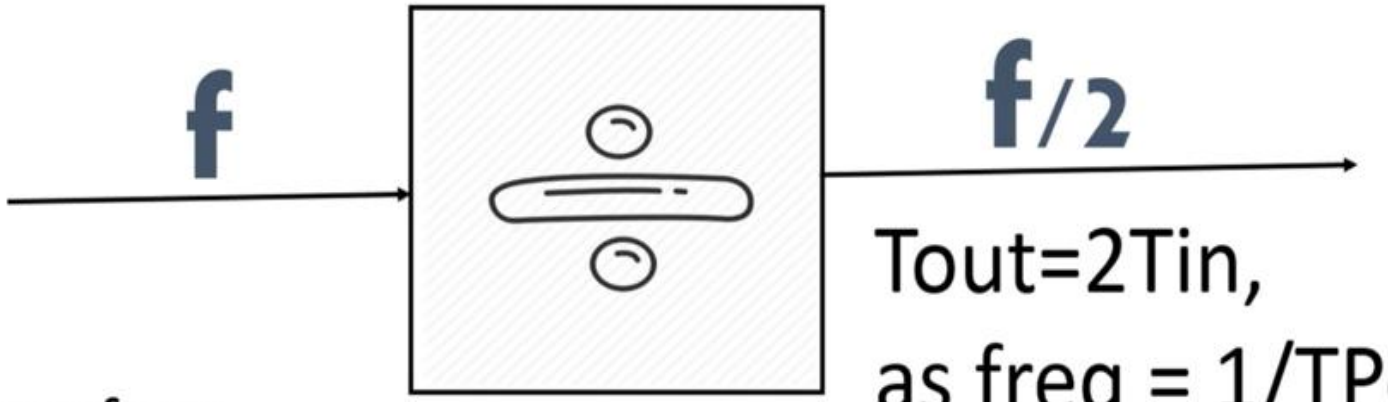
Optimization

Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0



Frequency Divider for f/n, where n is an even number

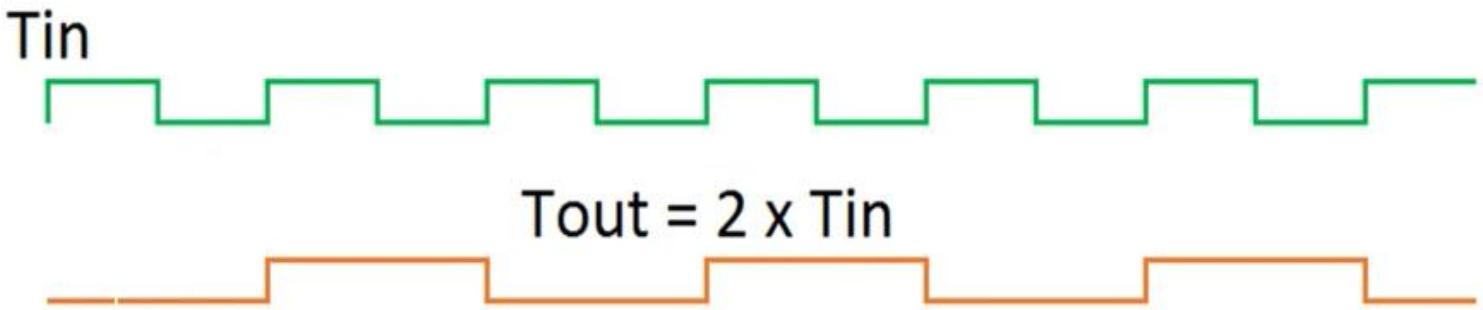
Let us consider the easiest case, n =2



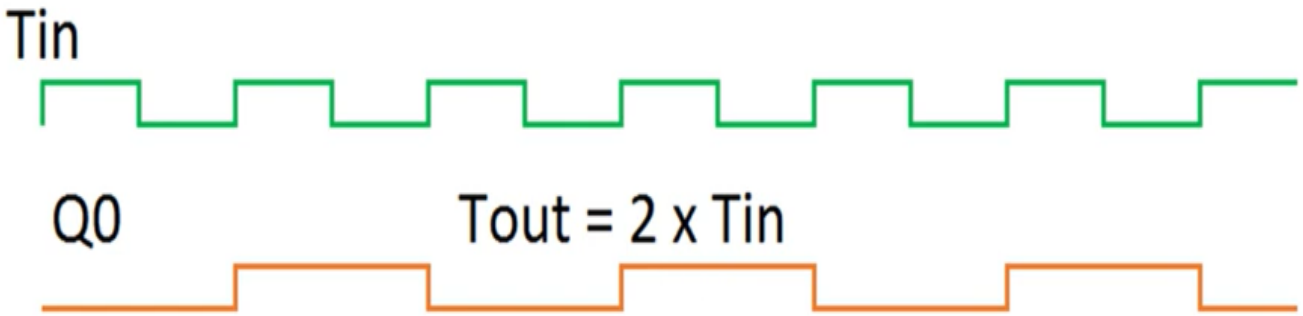
Tout=2Tin,

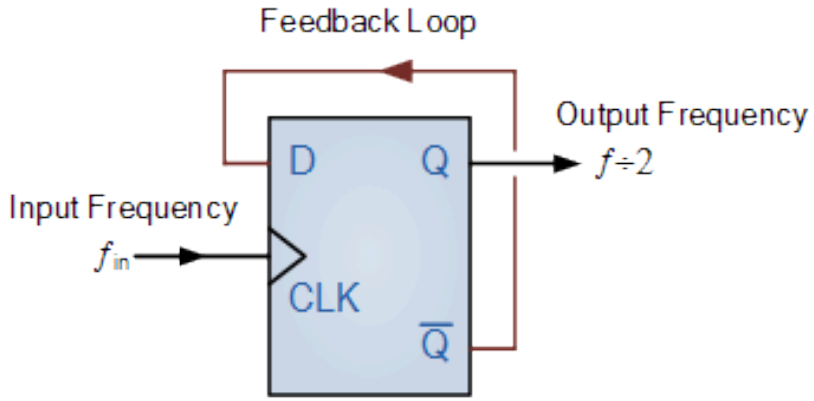
as freq = 1/TPeriod

Expected waveform

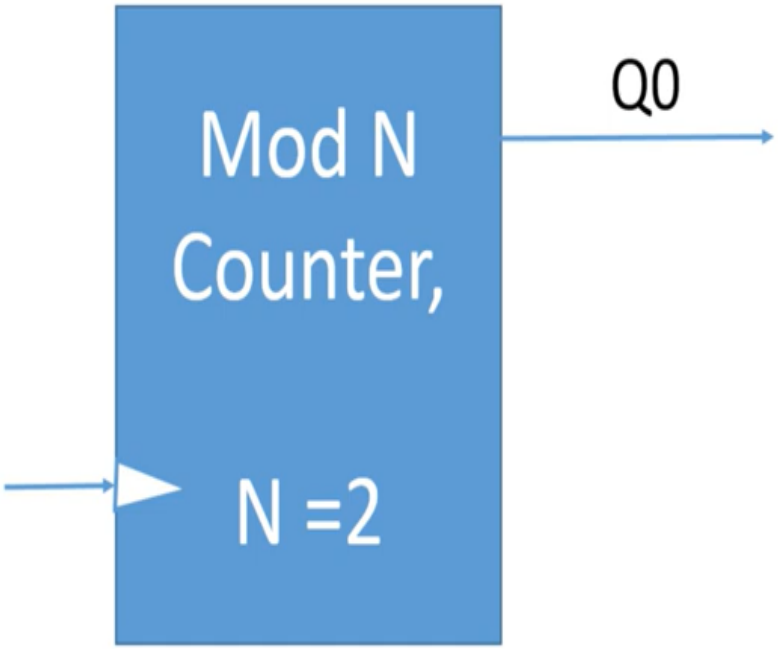


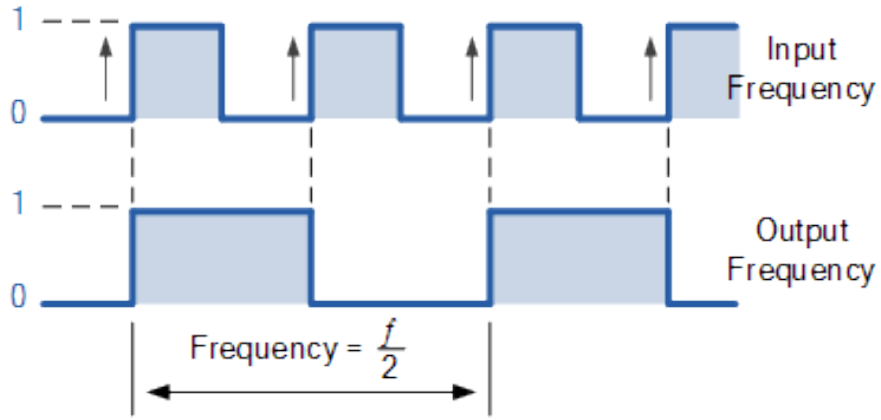
Frequency Divider for f/2



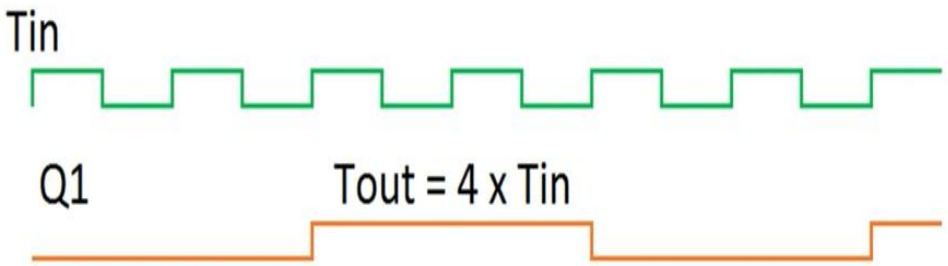


Q0
0
1

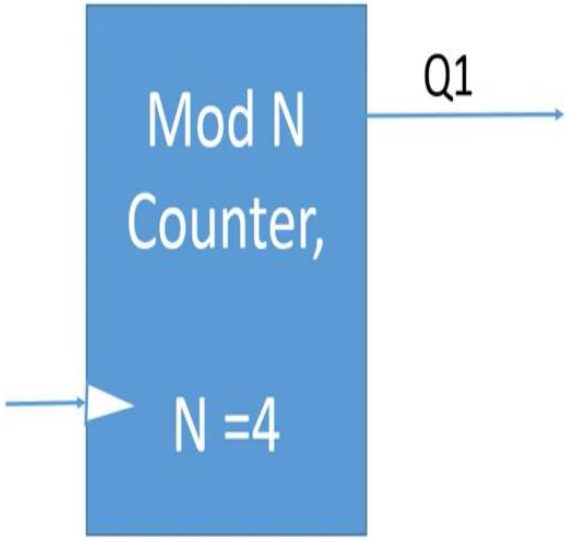




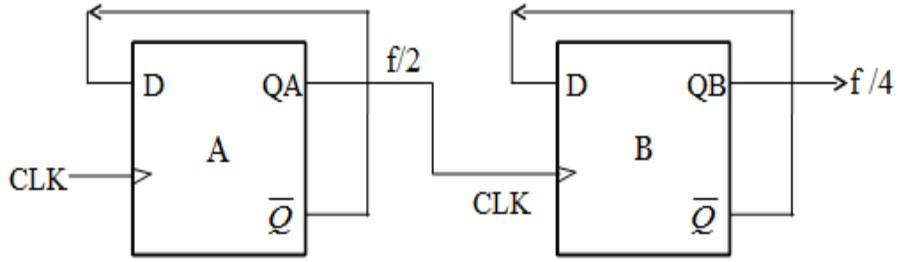
Frequency Divider for f/4

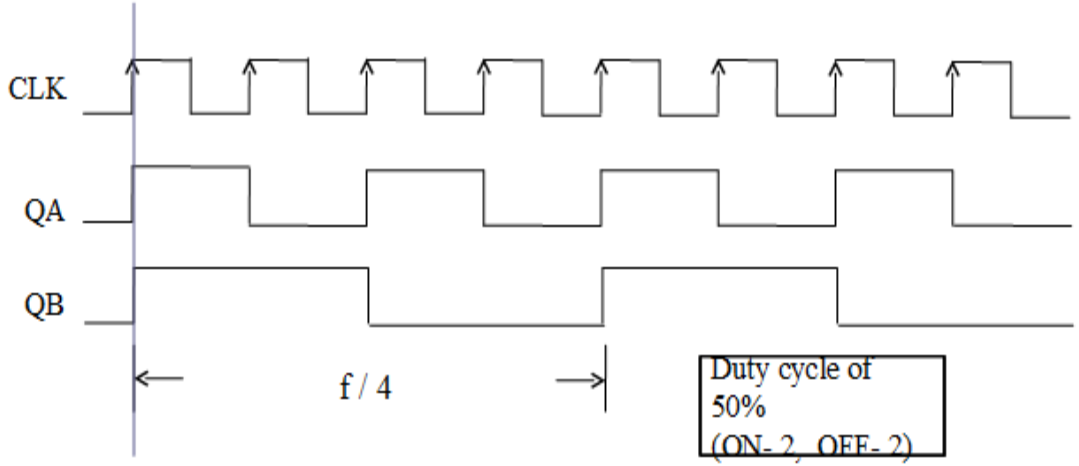


Q1	Q0
0	0
0	1
1	0
1	1



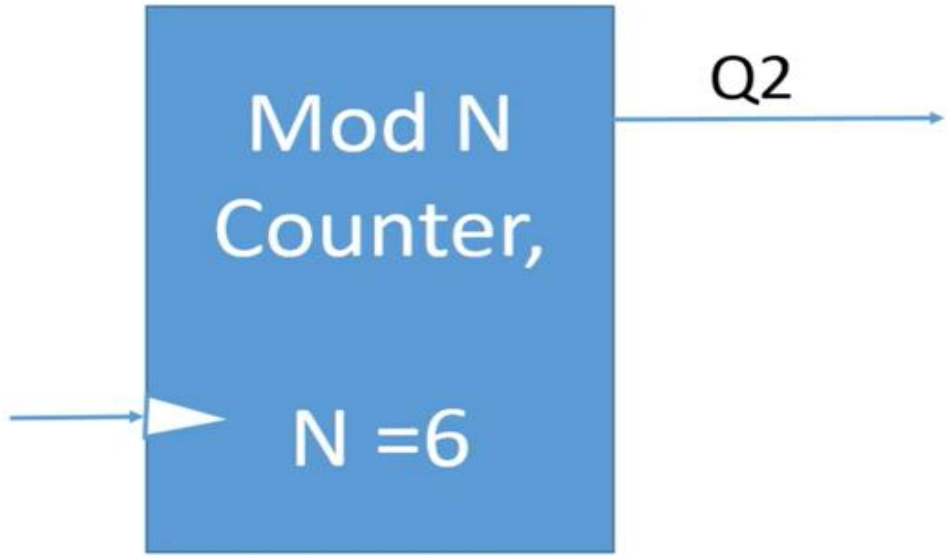
Frequency Divide by 4 (f/4)



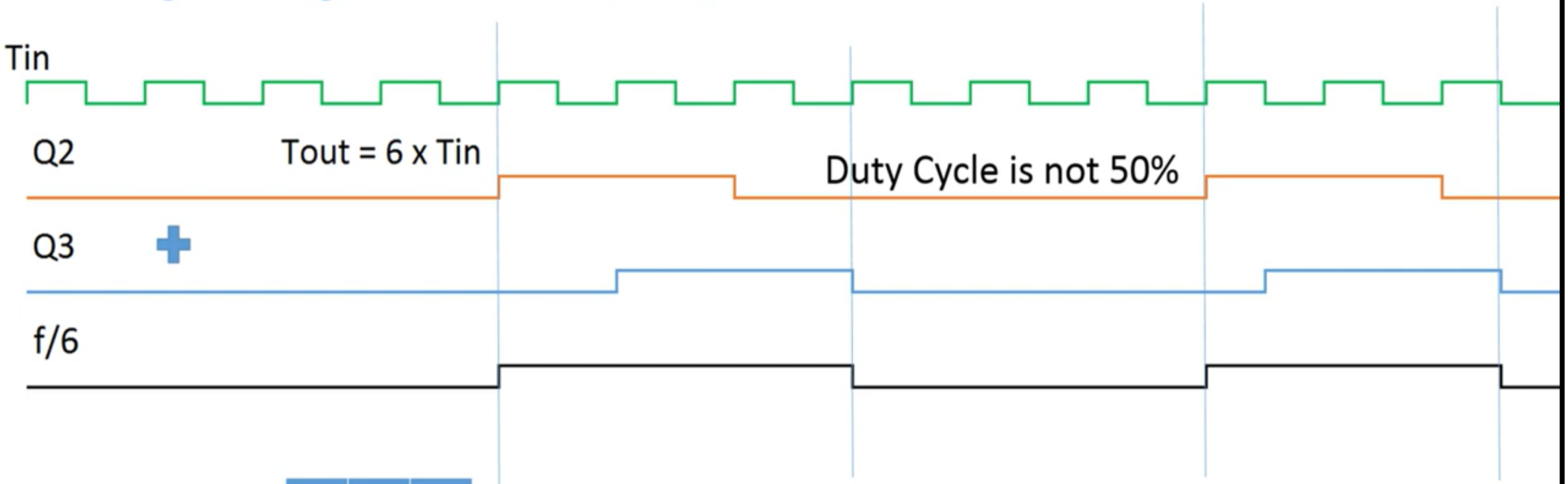


Frequency Divider for f/6

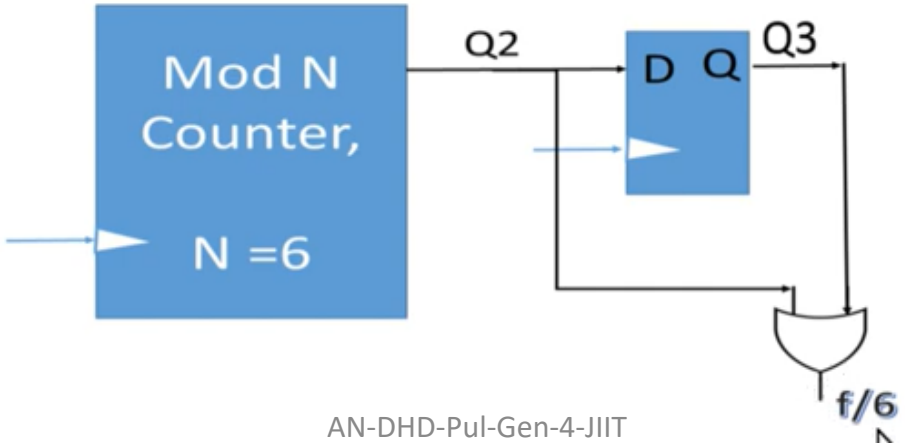
Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1



Frequency Divider for f/6

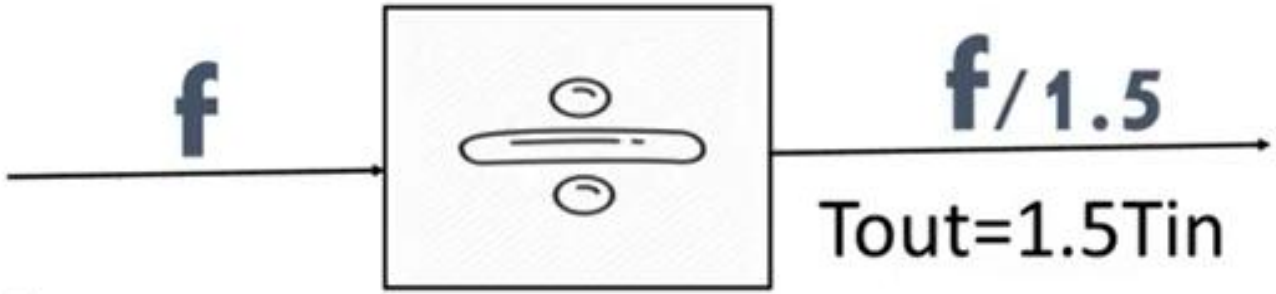


Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1

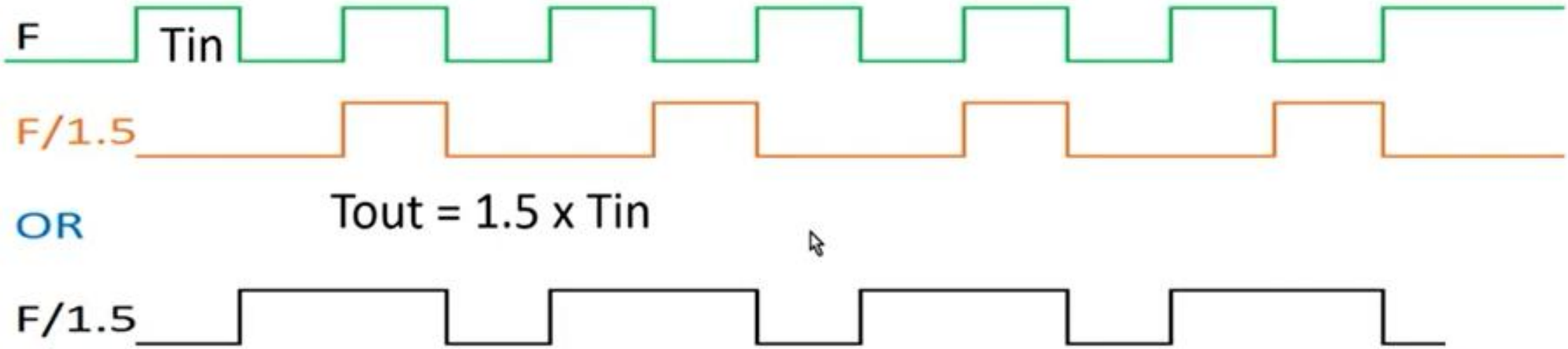


Frequency Divider for f/n, where n is a fraction

Let us consider the easiest case, n =1.5



Expected waveform

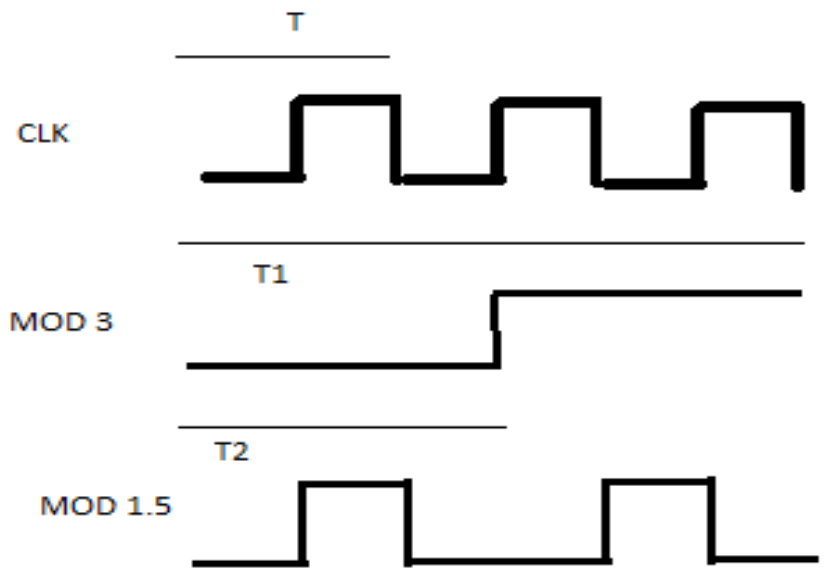


Clock Divider F/n n is integer

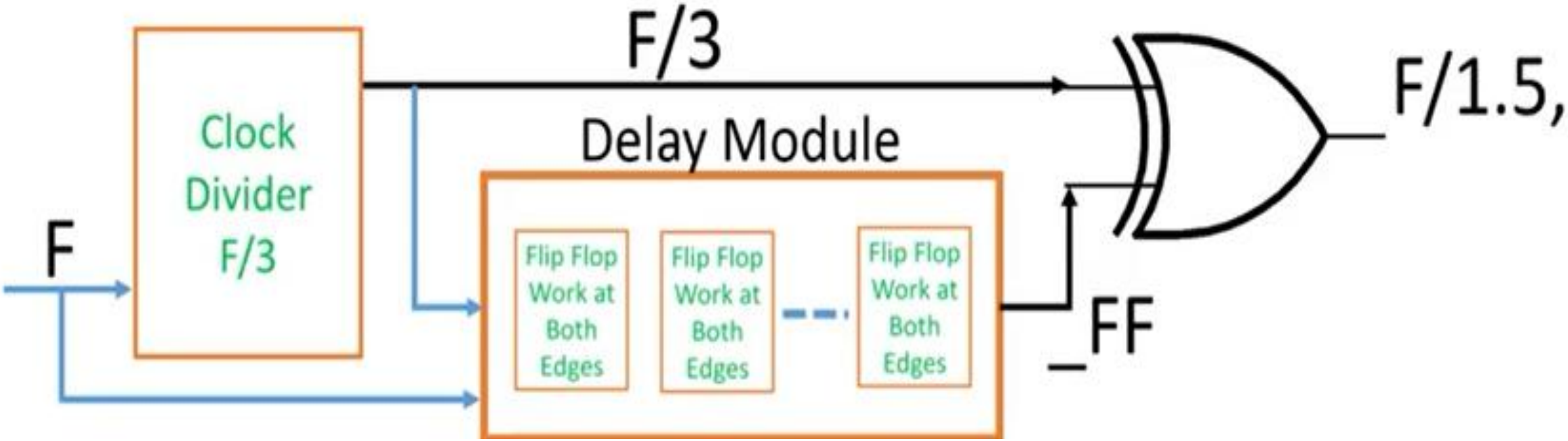
Flip Flop Work at Both Edges

T1= 3T so 1/F1= 3/f
F1=f/3

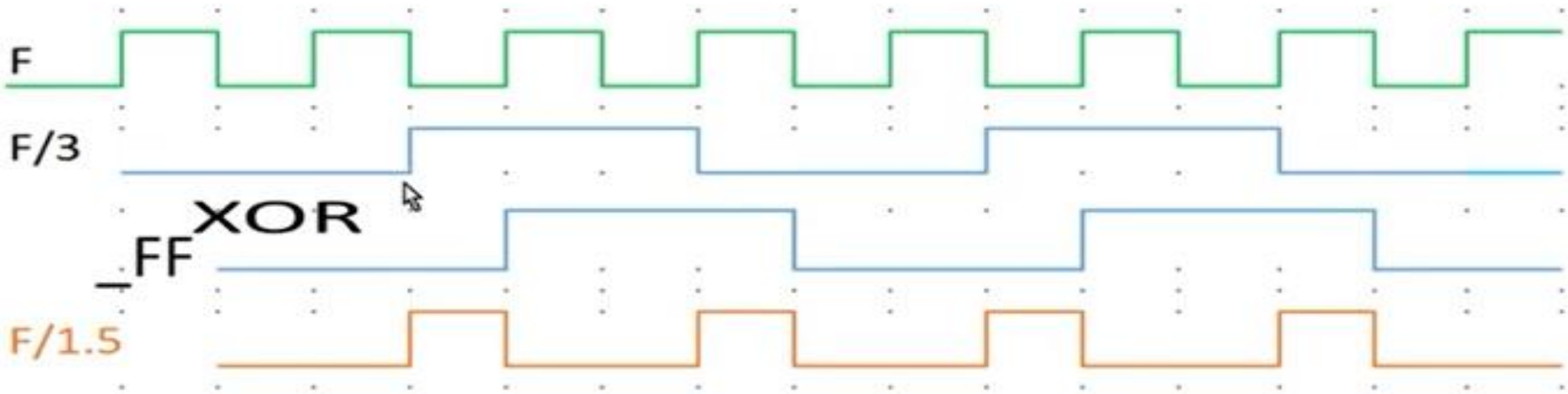
Similarly
T2=T1/2
So F2= 2F1=2*f/3= f/1.5

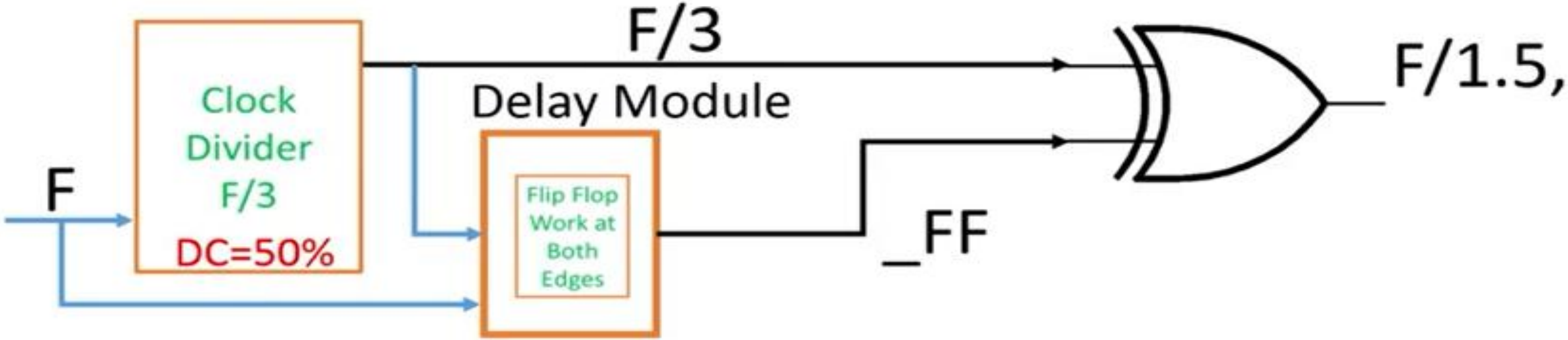


Frequency Divider for f/1.5

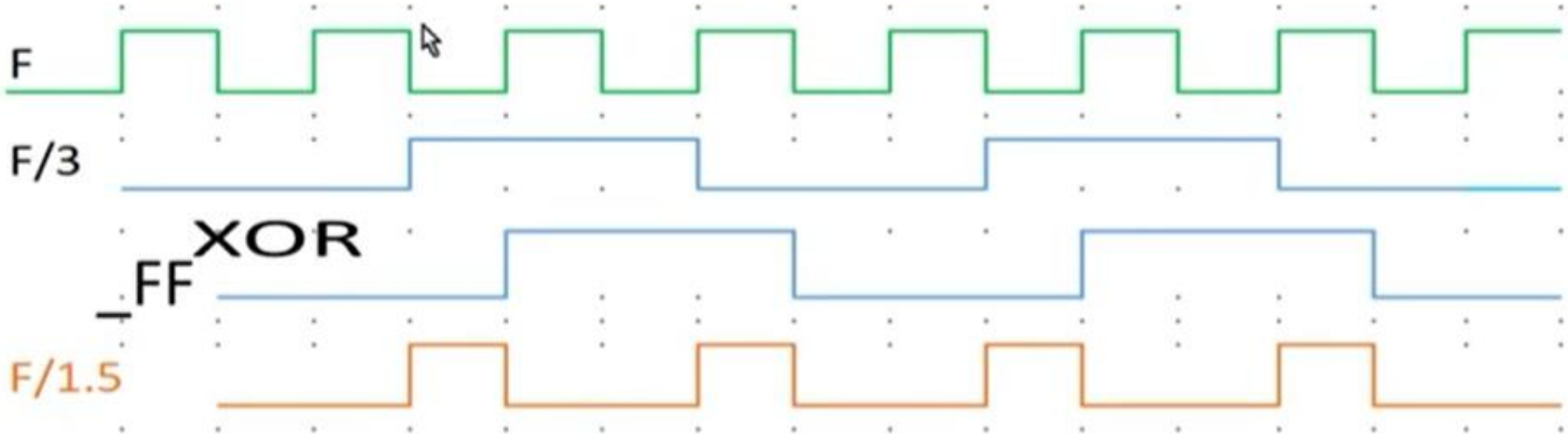


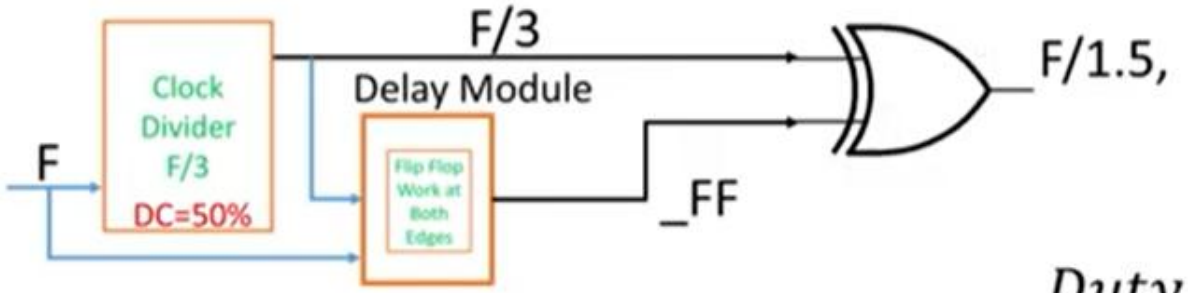
Frequency Divider for f/1.5





Frequency Divider for f/1.5





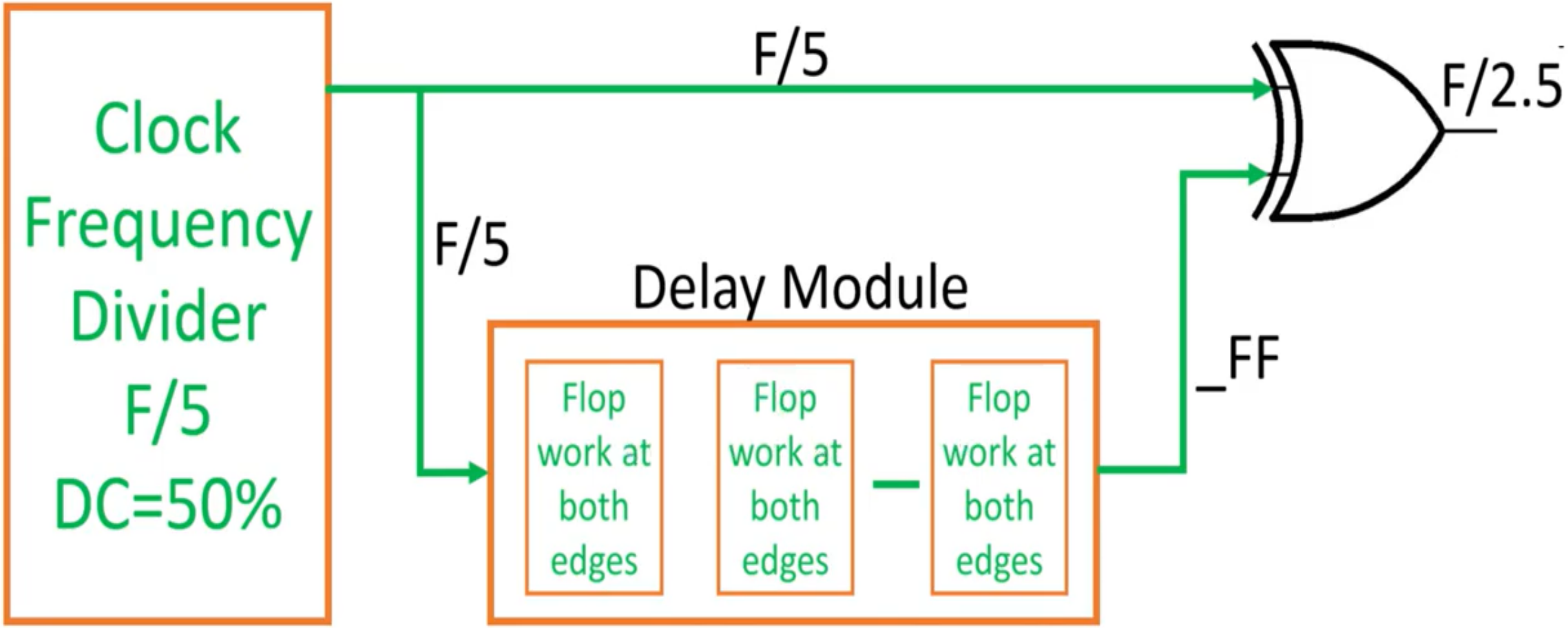
$$Duty\ Cycle = \frac{Ton}{Ton + Toff}$$

$$Duty\ Cycle = \frac{Tin/2}{Tin/2 + Tin} = \frac{1}{3}\ or\ 33.3\%$$

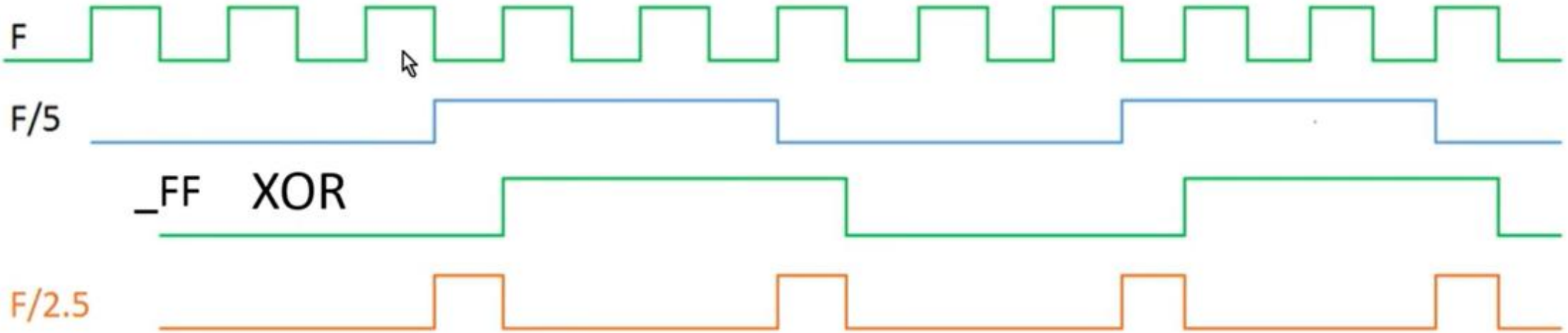
Conclusion for implementation of f/n divider

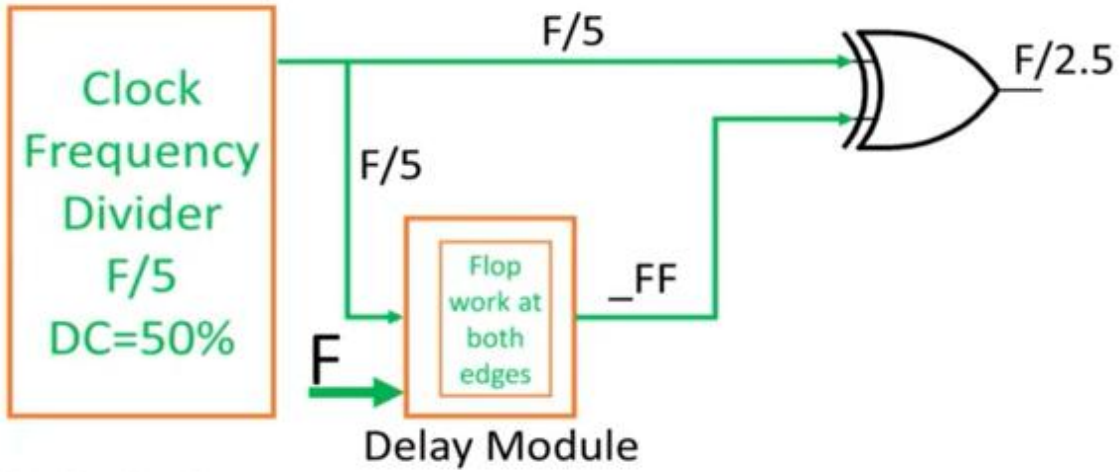
- Three building blocks are used to implement Frequency divider by a fractional number: Clock frequency divider by integer number, Delay Module and XOR gate.
- Multiply n by 2 and get the frequency (f/2n) to be generated by integer clock frequency divider. It is easy to generate f/2n as 2n is integer number.
- Depending upon the duty cycle, decide number of flip flops to be added in the delay module.

Frequency Divider for F/2.5



Frequency Divider for F/2.5

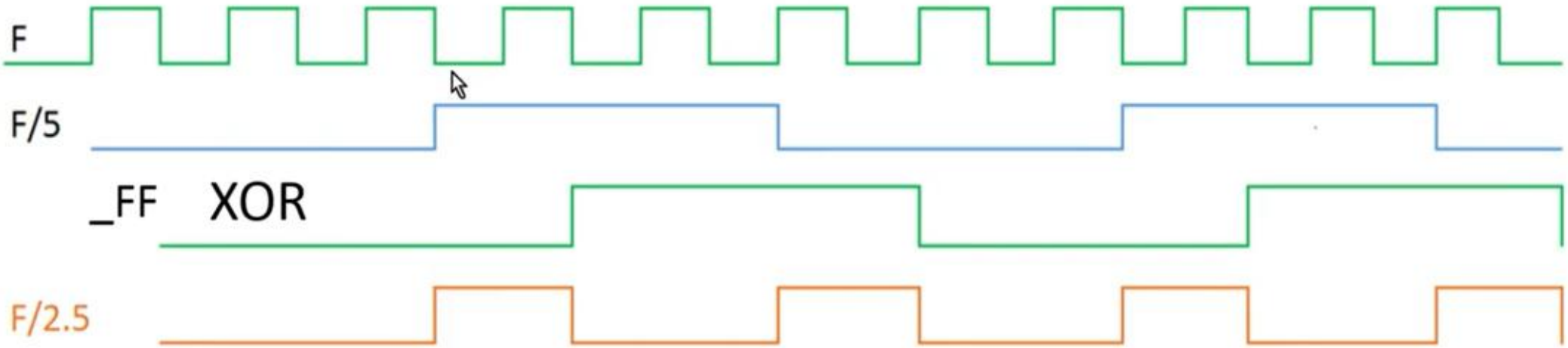


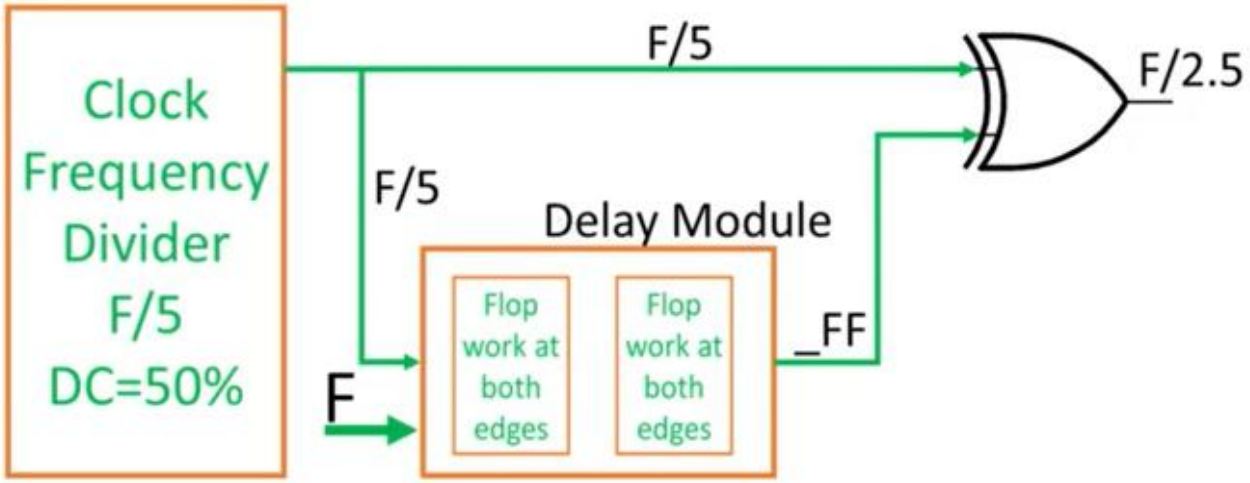


$$Duty\ Cycle = \frac{Ton}{Ton + Toff}$$

$$Duty\ Cycle = \frac{1}{5}\ or\ 20\%$$

Frequency Divider for F/2.5

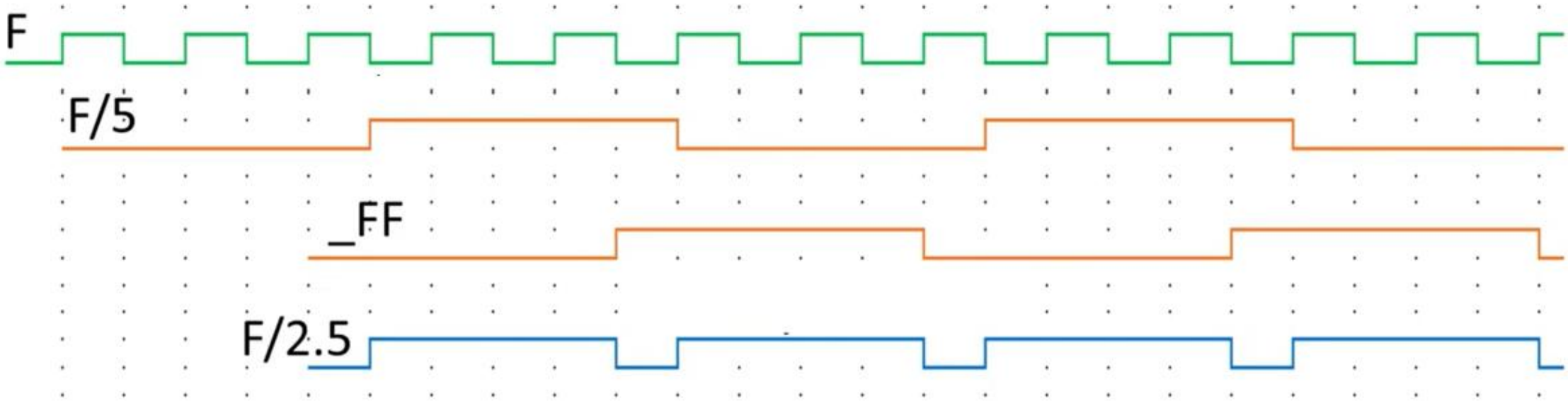


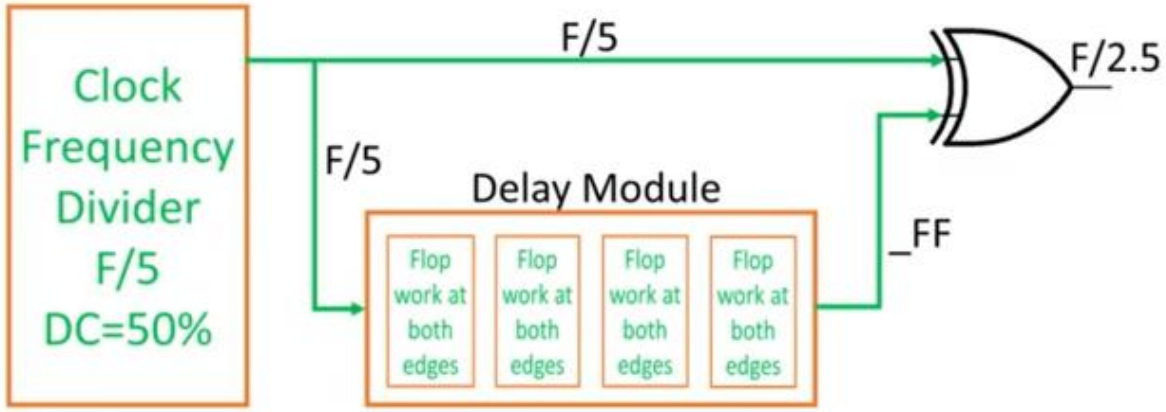


$$Duty\ Cycle = \frac{Ton}{Ton + Toff}$$

$$Duty\ Cycle = \frac{2}{5}\ or\ 40\%$$

Frequency Divider for F/2.5

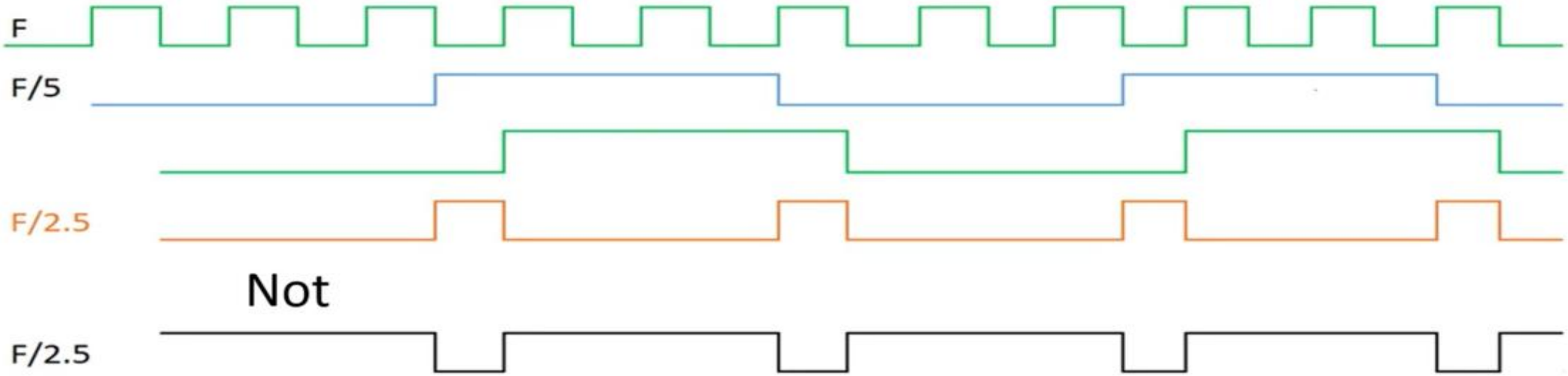


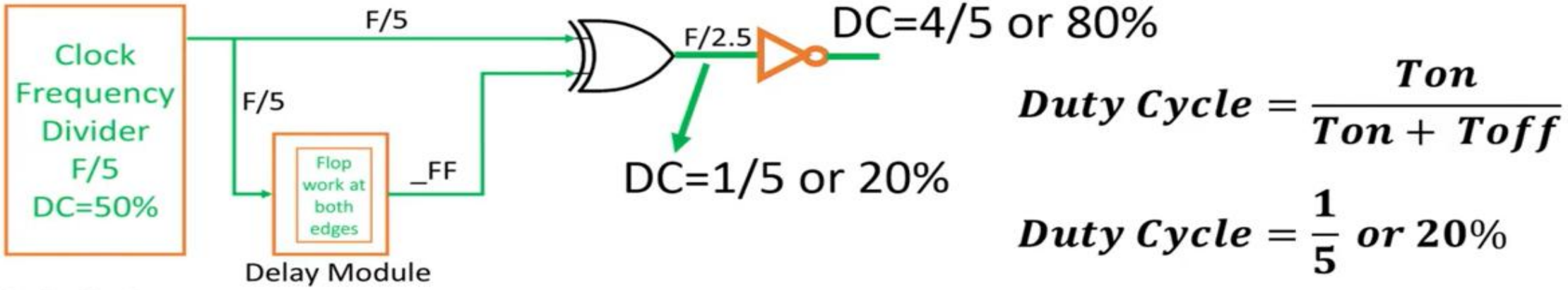


$$Duty\ Cycle = \frac{Ton}{Ton + Toff}$$

$$Duty\ Cycle = \frac{4}{5}\ or\ 80\%$$

Hardware optimization for F/2.5, DC = 4/5 or 80%





$$Duty\ Cycle = \frac{Ton}{Ton + Toff}$$

$$Duty\ Cycle = \frac{1}{5}\ or\ 20\%$$

Conclusion for implementation of f/n divider

- Four building blocks are used to implement Frequency divider by a fractional number: Clock frequency divider by integer number, Delay Module, XOR gate, XNOR gate.
- Multiply n by 2 and get the frequency (f/2n) to be generated by integer clock frequency divider. It is easy to generate f/2n as 2n is integer number.
- Depending upon the duty cycle, decide number of flip flops to be added in the delay module.
- Apply hardware optimization technique to reduce number of flip flops inside the delay module.