Mod 8 Up/Down Synchronous Counter using 130nm CMOS Technology

Swagatika Meher
Odisha University of Technology and Research, Bhubaneswar, Odisha, India

***Abstract*— In this study, 3 bit or mod 8 up/down synchronous counter is designed using transistors and Verilog code. This design is built with both analog and digital circuitry. As the counter is synchronous, the clock signal is applied to all of the flipflops at the same time. Here, an astable multivibrator is used to generate the clock signal. The analog circuitry consists of the astable multivibrator, the combination of AND gates and XOR gate implemented using CMOS transistors. The digital circuitry of MOD - 8 is made up of three T flipflops with synchronous clear. The control input M=0 is assumed for UP counting and M=1 for DOWN counting. The entire design process of sequential circuits and simulations are carried out using eSim software.**

Keywords— Synchronous counter, T flipflop, CMOS, Astable multivibrator

Circuit Details:-

In 3 bit or mod-8 Up or Down counting, 3 Flip Flops are required, which can count up to 2^3-1 = 7. Here, the counter's mode control input determines which sequence will be generated. In this scenario, the counter's mode control input determines whether it will execute up counting or down counting. Such a counter must be designed similarly to a synchronous counter, but it also needs additional combinational logic for mode control input.
Here 3 T flipflops are designed using Verilog code. For the control input (M), let's assume M=0 for UP counting and M=1 for DOWN counting.

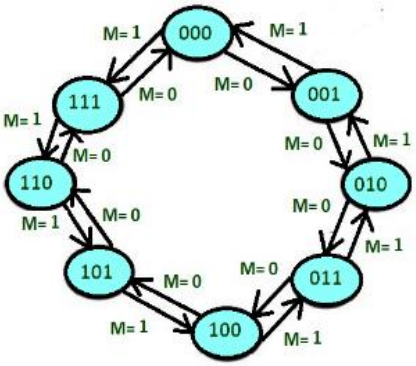


Fig.1: State transition diagram of 3 bit up/down counting

For the generation of clock pulse, the astable multivibrator is constructed by cascading three inverters, and the clock pulse signal is produced using a resistor and capacitor. The inverters serve as a buffer, and an important factor in switching the inputs and outputs of the inverter is the direction in which the capacitor is charged and discharged.

When the positive edge triggered clock pulse is applied and input T of the flip-flops, the present states of the counting sequence, as well as the next states are represented by the circuit excitation table. Using the Flip Flops excitation table, we can determine the input values for 3 Flip Flops by observing the change from one state to the subsequent state. In table 1, the excitation table is created using the necessary counting sequence.
Further, the circuit excitation table is reduced using the K-map to obtain the Boolean functions for the input to the flipflops. To create circuit schematics, the simplified expression for Flip Flops is adopted. In this case, all connections are formed using reduced expressions for flip flops.
The Boolean functions for input of flipflops are,
For

$$T_3 = M'Q_2Q_1 + MQ_2'Q_1'$$

$$T_2 = M'Q_1 + MQ_1'$$

$$T_1 = 1$$

Truth Table :-

The excitation table for each flipflop is shown below,

	Control Input (M)	Present State			Next State			Flipflops Inputs		
		Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+	T_3	T_2	T_1
Up Counting	0	0	0	0	0	0	1	0	0	1
	0	0	0	1	0	1	0	0	1	1
	0	0	1	0	0	1	1	0	0	1
	0	0	1	1	1	0	0	1	1	1
	0	1	0	0	1	0	1	0	0	1
	0	1	0	1	1	1	0	0	1	1
	0	1	1	0	1	1	1	0	0	1
	0	1	1	1	0	0	0	1	1	1
Down Counting	1	0	0	0	1	1	1	1	1	1
	1	0	0	1	0	0	0	0	0	1
	1	0	1	0	0	0	1	0	1	1
	1	0	1	1	0	1	0	0	0	1
	1	1	0	0	0	1	1	1	1	1
	1	1	0	1	1	0	0	0	0	1
	1	1	1	0	1	0	1	0	1	1
	1	1	1	1	1	1	0	0	0	1

Proposed Circuit diagram in eSim:-

The followings are the Schematic diagram drawn in eSim.

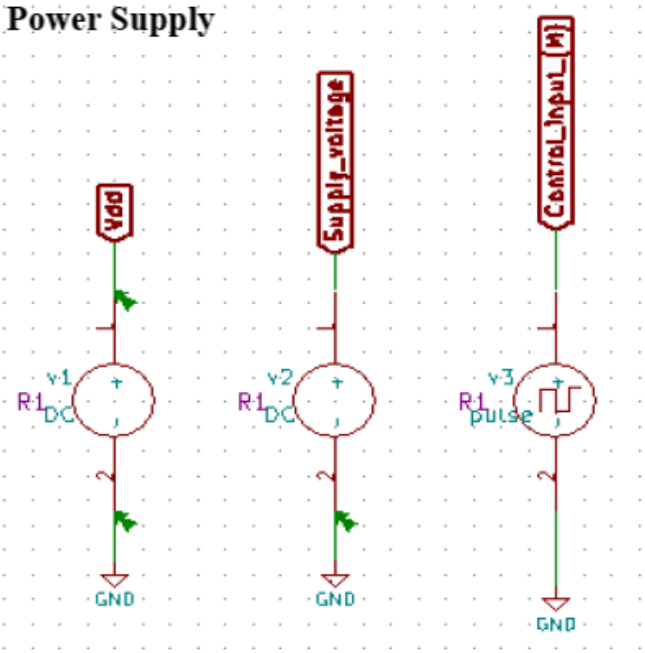


Fig.2: Power Supply of 3 bit up/down synchronous counter

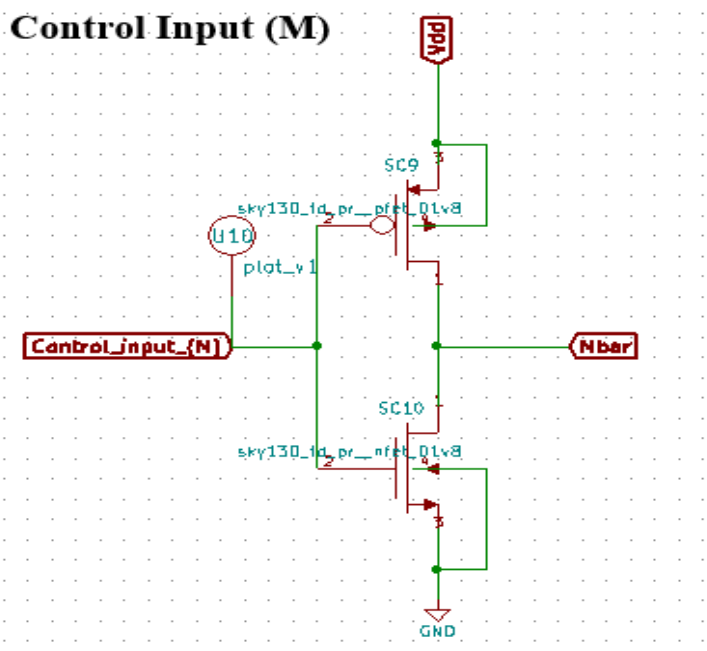


Fig.3: Control Input (M) of 3 bit up/down synchronous counter

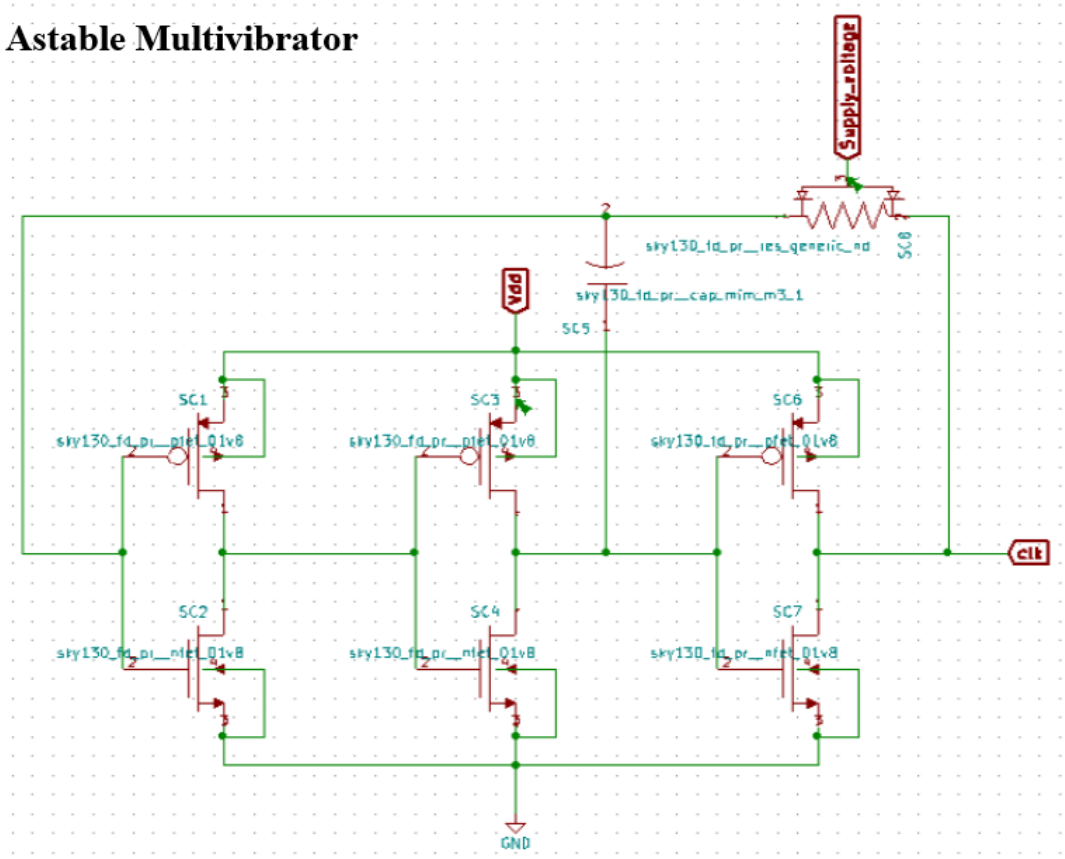


Fig.4: Astable Multivibrator for the generation of clock pulse

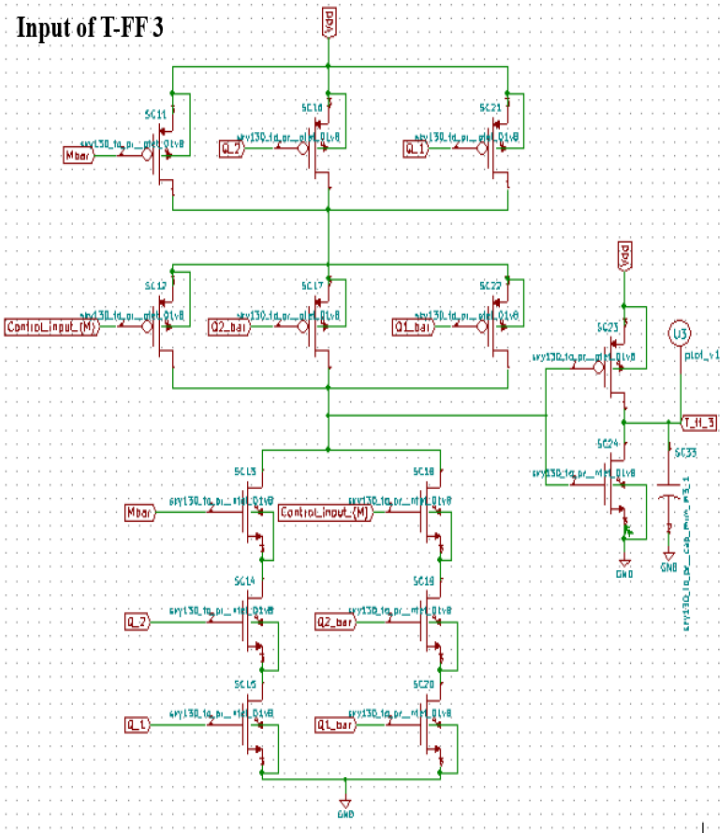


Fig.5: Input of T- flipflop 3 (T_3)

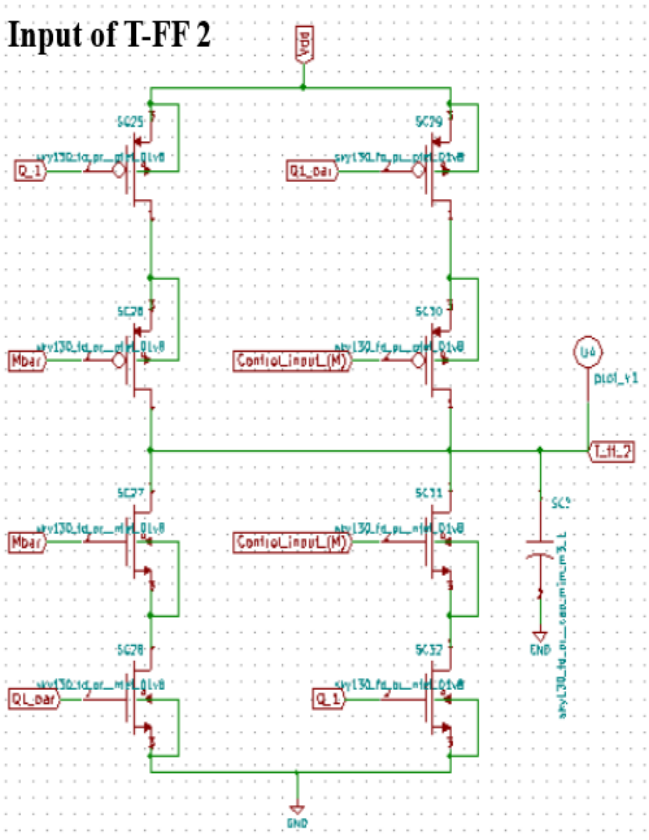


Fig.6: Input of T- flipflop 2 (T_2)

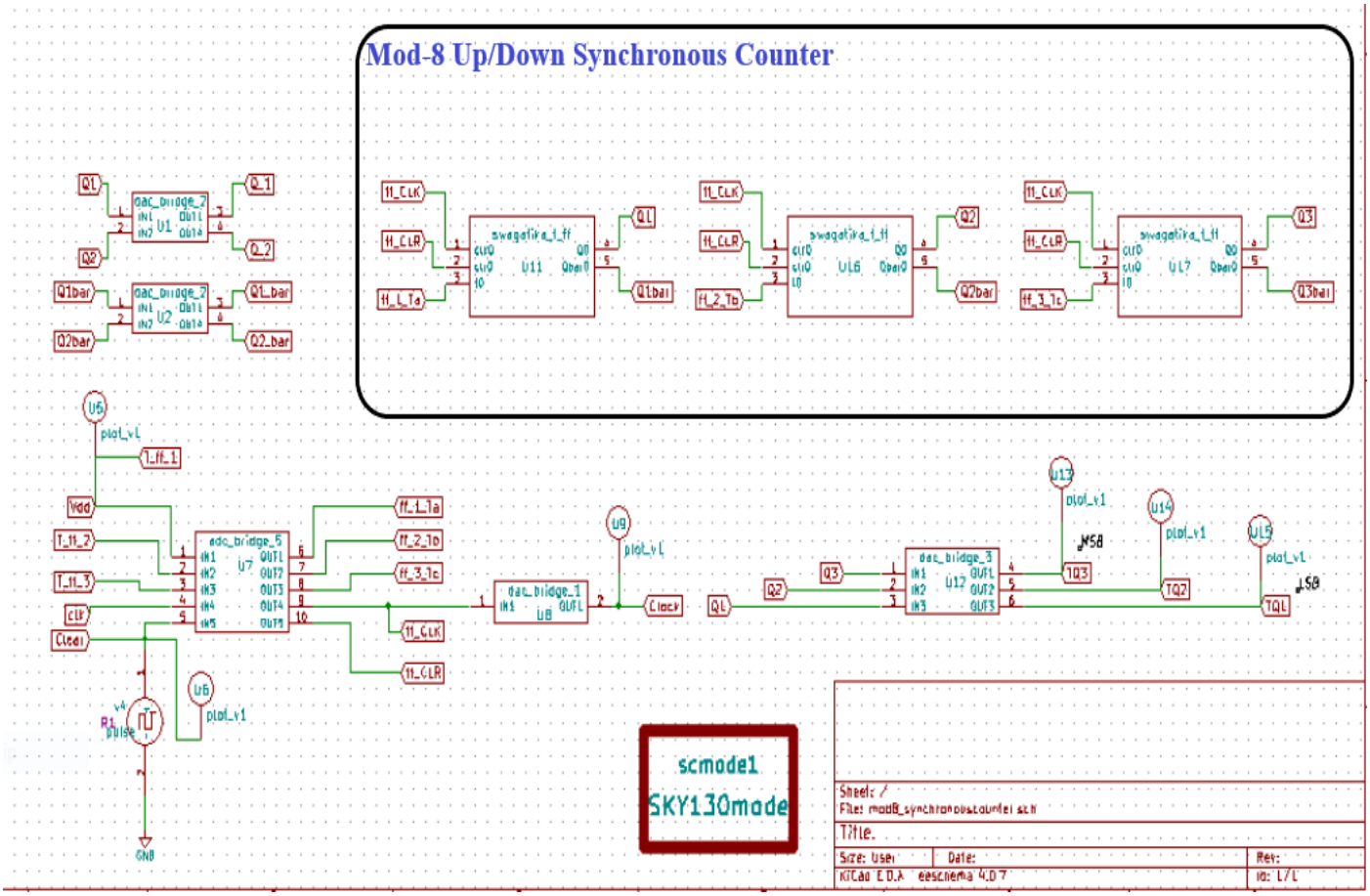


Fig.7: Schematic Diagram of MOD-8 UP/DOWN Synchronous counter using T-flipflops

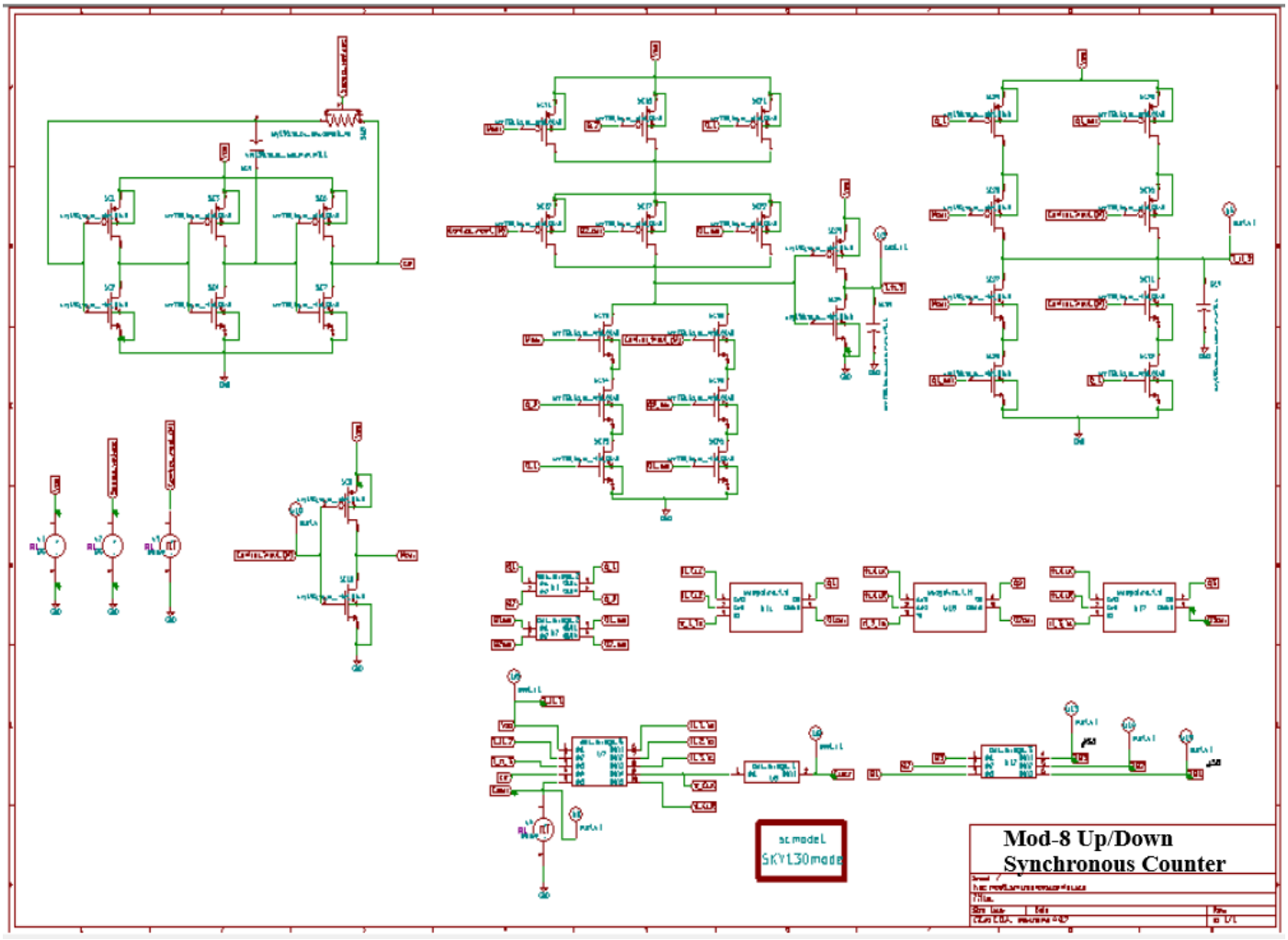


Fig.8: Mixed signal MOD-8 UP/DOWN Synchronous counter using 130nm CMOS technology

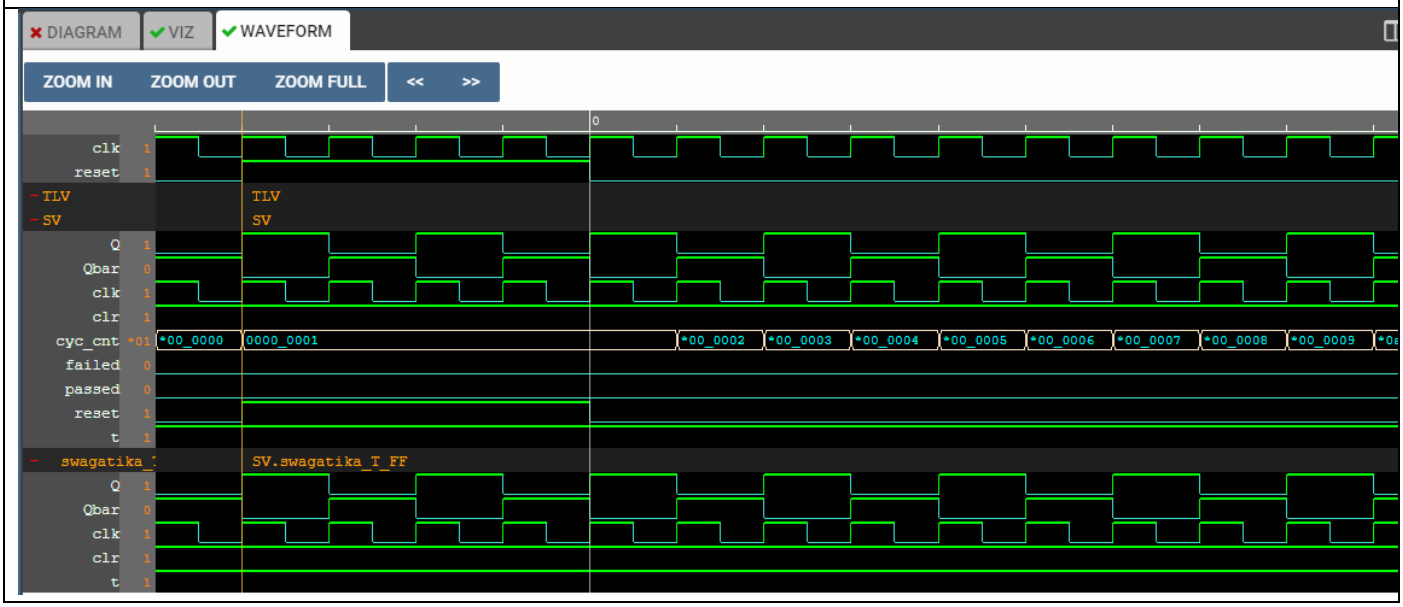
Verilog Code for T-Flipflop:-

```
//////////////////////////////////////////////////////////////////////
//
// Design Name: T Flipflop
// Designer: Swagatika Meher
// Module Name: swagatika_T_FF
//
//////////////////////////////////////////////////////////////////////

module swagatika_T_FF(t,clk,clr,Q,Qbar);
input clk,clr,t;
output reg Q;
output Qbar;
always @(posedge clk)
begin
if (!clr)
Q<=0;
else
if (t)
Q<=~Q;
else
Q<=Q;
end
assign Qbar = ~Q;
endmodule
```

Makerchip Plots for T-Flipflop:-

```
\TLV_version 1d: tl-x.org
\SV
/* verilator lint_off UNUSED*/  /* verilator lint_off DECLFILENAME*/  /* verilator lint_off BLKSEQ*/  /* verilator lint_off WIDTH*/  /* verilator lint_off SELRANGE*/  /* verilator lint_off PINCONNECTEMPTY*/  /* verilator lint_off DEFPARAM*/  /* verilator lint_off IMPLICIT*/  /* verilator lint_off COMBDLY*/  /* verilator lint_off SYNCASYNCNET*/  /* verilator lint_off UNOPTFLAT */  /* verilator lint_off UNSIGNED*/  /* verilator lint_off CASEINCOMPLETE*/  /* verilator lint_off UNDRIVEN*/  /* verilator lint_off VARHIDDEN*/  /* verilator lint_off CASEX*/  /* verilator lint_off CASEOVERLAP*/  /* verilator lint_off PINMISSING*/  /* verilator lint_off BLKANDNBLK*/  /* verilator lint_off MULTIDRIVEN*/  /* verilator lint_off WIDTHCONCAT*/  /* verilator lint_off ASSIGNDLY*/  /* verilator lint_off MODDUP*/  /* verilator lint_off STMTDLY*/  /* verilator lint_off LITENDIAN*/  /* verilator lint_off INITIALDLY*/

//Your Verilog/System Verilog Code Starts Here:
//////////////////////////////////////////////////////////////////////
//
// Design Name:       T Flipflop
// Designer:          Swagatika Meher
// Module Name:       swagatika_T_FF
//
//////////////////////////////////////////////////////////////////////


module swagatika_T_FF(t,clk,clr,Q,Qbar);
        input clk,clr,t;
        output reg Q;
        output Qbar;

        always @(posedge clk)
        begin
                if (!clr)
                        Q<=0;
                else
```

```
                        if (t)
                                Q<=~Q;
                        else
                                                Q<=Q;
        end
                assign Qbar = ~Q;
endmodule



//Top Module Code Starts here:
        module top(input logic clk, input logic reset, input logic [31:0] cyc_cnt, output logic passed, output logic failed);
                logic  clr;//input
                logic  t;//input
                logic  Q;//output
                logic  Qbar;//output
//The $random() can be replaced if user wants to assign values
                assign clr = $random();
                assign t = $random();
                swagatika_T_FF swagatika_T_FF(.clk(clk), .clr(clr), .t(t), .Q(Q), .Qbar(Qbar));

\TLV
//Add \TLV here if desired
\SV
endmodule
```



Netlists and NgSpice plots for mod-8 up/down synchronous counter:-

```
* e:\esim\mod8_synchronouscounter\mod8_synchronouscounter.cir

.include "C:\FOSSEE\eSim\library\sky130_fd_pr\models\sky130_fd_pr__model__r+c.model.spice"
.include "C:\FOSSEE\eSim\library\sky130_fd_pr\models\sky130_fd_pr__model__linear.model.spice"
.lib "C:\FOSSEE\eSim\library\sky130_fd_pr\models\sky130.lib.spice" tt
.include
"C:\FOSSEE\eSim\library\sky130_fd_pr\models\sky130_fd_pr__model__diode_pd2nw_11v0.model.spic
e"
.include "C:\FOSSEE\eSim\library\sky130_fd_pr\models\sky130_fd_pr__model__pnp.model.spice"
.include "C:\FOSSEE\eSim\library\sky130_fd_pr\models\sky130_fd_pr__model__inductors.model.spice"
.include
"C:\FOSSEE\eSim\library\sky130_fd_pr\models\sky130_fd_pr__model__diode_pw2nd_11v0.model.spic
e"
v1 t_ff_1 gnd dc 1.8
```

```
xsc1 net-_sc1-pad1_ net-_sc1-pad2_ t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc3 net-_sc3-pad1_ net-_sc1-pad1_ t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc6 clk net-_sc3-pad1_ t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc2 net-_sc1-pad1_ net-_sc1-pad2_ gnd gnd sky130_fd_pr__nfet_01v8
xsc4 net-_sc3-pad1_ net-_sc1-pad1_ gnd gnd sky130_fd_pr__nfet_01v8
xsc7 clk net-_sc3-pad1_ gnd gnd sky130_fd_pr__nfet_01v8
xsc5 net-_sc3-pad1_ net-_sc1-pad2_ sky130_fd_pr__cap_mim_m3_1 W=1000 L=6000 MF=5000
xsc8 net-_sc1-pad2_ clk supply_voltage sky130_fd_pr__res_generic_nd W=1 L=900
v2 supply_voltage gnd dc 3.5
* s c m o d e
xsc9 mbar control_input__m_ t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc10 mbar control_input__m_ gnd gnd sky130_fd_pr__nfet_01v8
v3 control_input__m_ gnd pulse(3.5 0 0 0.001m 0.001m 50m 100m)
* u2 q1bar q2bar q1_bar q2_bar dac_bridge_2
* u1 q1 q2 q_1 q_2 dac_bridge_2
xsc11 net-_sc11-pad1_ mbar t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc16 net-_sc11-pad1_ q_2 t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc21 net-_sc11-pad1_ q_1 t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc12 net-_sc12-pad1_ control_input__m_ net-_sc11-pad1_ net-_sc11-pad1_ sky130_fd_pr__pfet_01v8
xsc17 net-_sc12-pad1_ q2_bar net-_sc11-pad1_ net-_sc11-pad1_ sky130_fd_pr__pfet_01v8
xsc22 net-_sc12-pad1_ q1_bar net-_sc11-pad1_ net-_sc11-pad1_ sky130_fd_pr__pfet_01v8
xsc13 net-_sc12-pad1_ mbar net-_sc13-pad3_ net-_sc13-pad3_ sky130_fd_pr__nfet_01v8
xsc14 net-_sc13-pad3_ q_2 net-_sc14-pad3_ net-_sc14-pad3_ sky130_fd_pr__nfet_01v8
xsc15 net-_sc14-pad3_ q_1 gnd gnd sky130_fd_pr__nfet_01v8
xsc18 net-_sc12-pad1_ control_input__m_ net-_sc18-pad3_ net-_sc18-pad3_ sky130_fd_pr__nfet_01v8
xsc19 net-_sc18-pad3_ q2_bar net-_sc19-pad3_ net-_sc19-pad3_ sky130_fd_pr__nfet_01v8
xsc20 net-_sc19-pad3_ q1_bar gnd gnd sky130_fd_pr__nfet_01v8
xsc23 t_ff_3 net-_sc12-pad1_ t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc24 t_ff_3 net-_sc12-pad1_ gnd gnd sky130_fd_pr__nfet_01v8
* u3 t_ff_3 plot_v1
xsc25 net-_sc25-pad1_ q_1 t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc26 t_ff_2 mbar net-_sc25-pad1_ net-_sc25-pad1_ sky130_fd_pr__pfet_01v8
xsc29 net-_sc29-pad1_ q1_bar t_ff_1 t_ff_1 sky130_fd_pr__pfet_01v8
xsc30 t_ff_2 control_input__m_ net-_sc29-pad1_ net-_sc29-pad1_ sky130_fd_pr__pfet_01v8
xsc27 t_ff_2 mbar net-_sc27-pad3_ net-_sc27-pad3_ sky130_fd_pr__nfet_01v8
xsc28 net-_sc27-pad3_ q1_bar gnd gnd sky130_fd_pr__nfet_01v8
xsc31 t_ff_2 control_input__m_ net-_sc31-pad3_ net-_sc31-pad3_ sky130_fd_pr__nfet_01v8
xsc32 net-_sc31-pad3_ q_1 gnd gnd sky130_fd_pr__nfet_01v8
* u4 t_ff_2 plot_v1
* u7 t_ff_1 t_ff_2 t_ff_3 clk clear ff_1_ta ff_2_tb ff_3_tc ff_clk ff_clr adc_bridge_5
* u5 t_ff_1 plot_v1
v4 clear gnd pulse(0 3.5 0 0.001m 0.001m 30m 50m)
* u6 clear plot_v1
* u8 ff_clk clock dac_bridge_1
* u9 clock plot_v1
* u11 ff_clk ff_clr ff_1_ta q1 q1bar swagatika_t_ff
* u16 ff_clk ff_clr ff_2_tb q2 q2bar swagatika_t_ff
* u17 ff_clk ff_clr ff_3_tc q3 ? swagatika_t_ff
* u12 q3 q2 q1 tq3 tq2 tq1 dac_bridge_3
* u13 tq3 plot_v1
* u14 tq2 plot_v1
* u15 tq1 plot_v1
* u10 control_input__m_ plot_v1
xsc33 t_ff_3 gnd sky130_fd_pr__cap_mim_m3_1 W=100 L=500 MF=5000
xsc34 t_ff_2 gnd sky130_fd_pr__cap_mim_m3_1 W=100 L=100 MF=5000
```

```
a1 [q1bar q2bar ] [q1_bar q2_bar ] u2
a2 [q1 q2 ] [q_1 q_2 ] u1
a3 [t_ff_1 t_ff_2 t_ff_3 clk clear ] [ff_1_ta ff_2_tb ff_3_tc ff_clk ff_clr ] u7
a4 [ff_clk ] [clock ] u8
a5 [ff_clk ] [ff_clr ] [ff_1_ta ] [q1 ] [q1bar ] u11
a6 [ff_clk ] [ff_clr ] [ff_2_tb ] [q2 ] [q2bar ] u16
a7 [ff_clk ] [ff_clr ] [ff_3_tc ] [q3 ] [? ] u17
a8 [q3 q2 q1 ] [tq3 tq2 tq1 ] u12
* Schematic Name: dac_bridge_2, NgSpice Name: dac_bridge
.model u2 dac_bridge(out_low=0.0 out_high=5.0 out_undef=0.5 input_load=1.0e-12 t_rise=1.0e-9
t_fall=1.0e-9 )
* Schematic Name: dac_bridge_2, NgSpice Name: dac_bridge
.model u1 dac_bridge(out_low=0.0 out_high=5.0 out_undef=0.5 input_load=1.0e-12 t_rise=1.0e-9
t_fall=1.0e-9 )
* Schematic Name: adc_bridge_5, NgSpice Name: adc_bridge
.model u7 adc_bridge(in_low=1.0 in_high=2.0 rise_delay=1.0e-9 fall_delay=1.0e-9 )
* Schematic Name: dac_bridge_1, NgSpice Name: dac_bridge
.model u8 dac_bridge(out_low=0.0 out_high=5.0 out_undef=0.5 input_load=1.0e-12 t_rise=1.0e-9
t_fall=1.0e-9 )
* Schematic Name: swagatika_t_ff, NgSpice Name: swagatika_t_ff
.model u11 swagatika_t_ff(rise_delay=1.0e-9 fall_delay=1.0e-9 input_load=1.0e-12 instance_id=1 )
* Schematic Name: swagatika_t_ff, NgSpice Name: swagatika_t_ff
.model u16 swagatika_t_ff(rise_delay=1.0e-9 fall_delay=1.0e-9 input_load=1.0e-12 instance_id=1 )
* Schematic Name: swagatika_t_ff, NgSpice Name: swagatika_t_ff
.model u17 swagatika_t_ff(rise_delay=1.0e-9 fall_delay=1.0e-9 input_load=1.0e-12 instance_id=1 )
* Schematic Name: dac_bridge_3, NgSpice Name: dac_bridge
.model u12 dac_bridge(out_low=0.0 out_high=5.0 out_undef=0.5 input_load=1.0e-12 t_rise=1.0e-9
t_fall=1.0e-9 )
.tran 0.01e-03 100e-03 0e-03

* Control Statements
.control
run
print allv > plot_data_v.txt
print alli > plot_data_i.txt
plot v(clock)+54 v(clear)+48 v(control_input__m_)+40 v(t_ff_3)+34 v(t_ff_2)+29 v(t_ff_1)+24
v(tq3)+17 v(tq2)+8 v(tq1)
.endc
.end
```

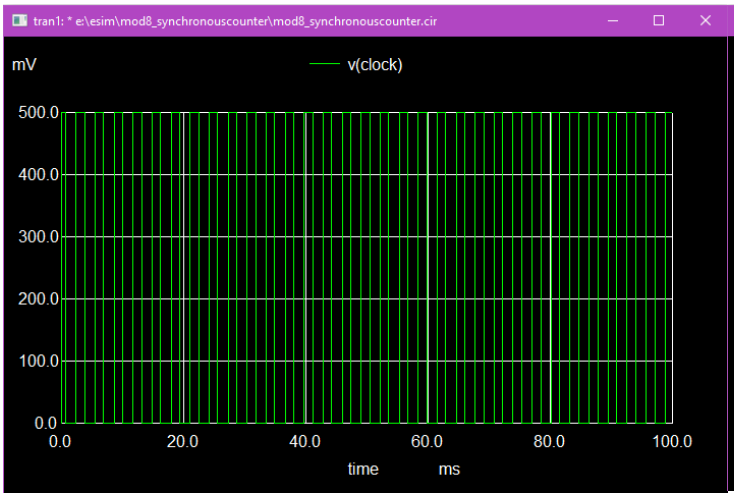


Fig.9: Clock Pulse

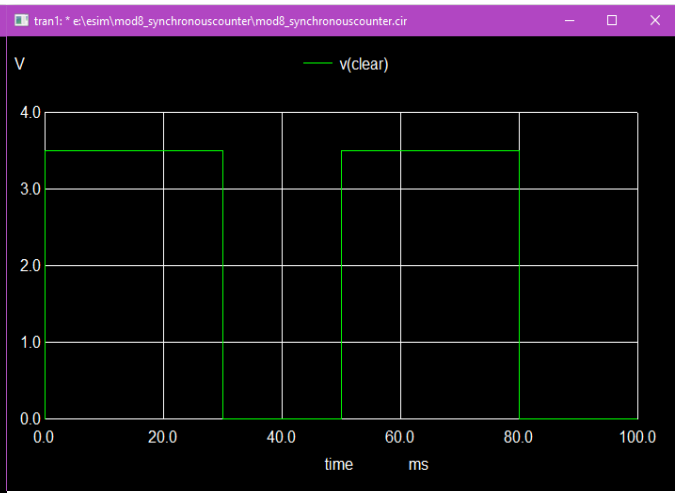


Fig.10: Clear Signal

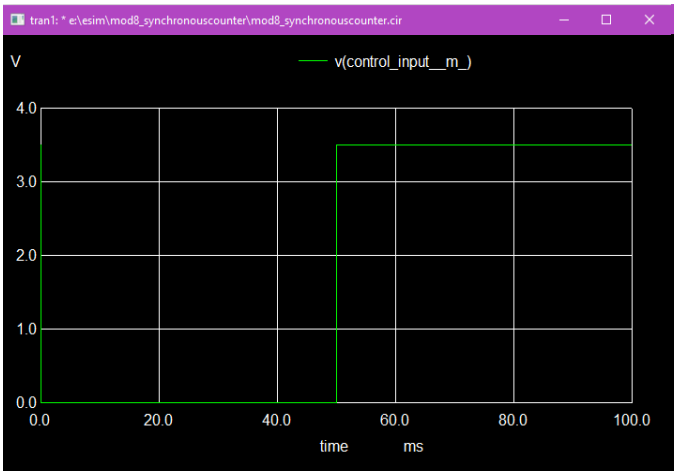


Fig.11: Control Input signal (M)

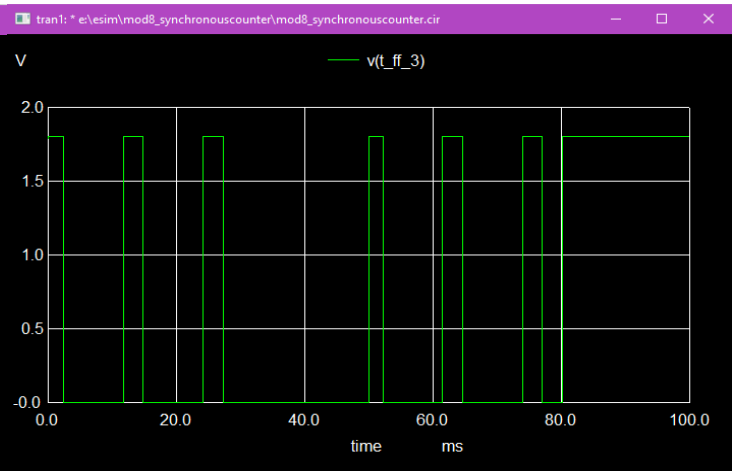


Fig.12: Input Signal of T-FF 3

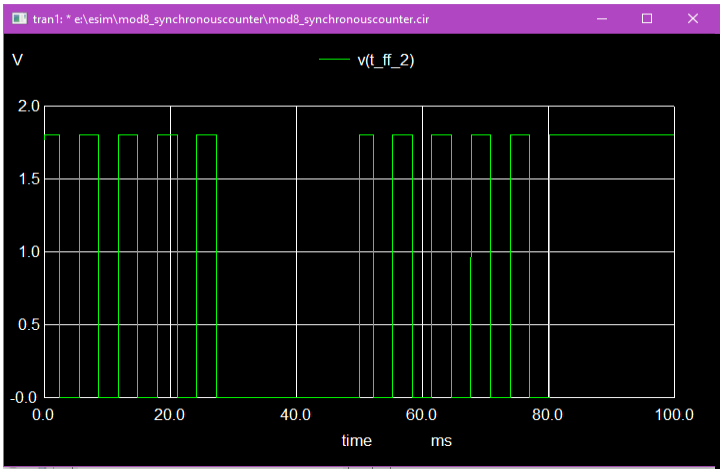


Fig.13: Input signal of T-FF 2

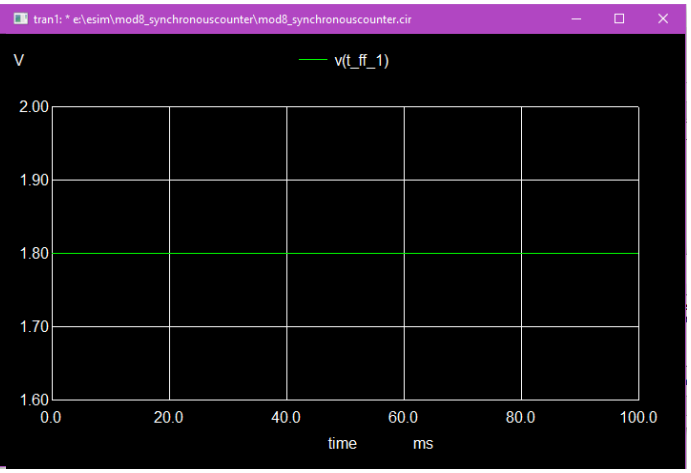


Fig.14: Input Signal of T-FF 1

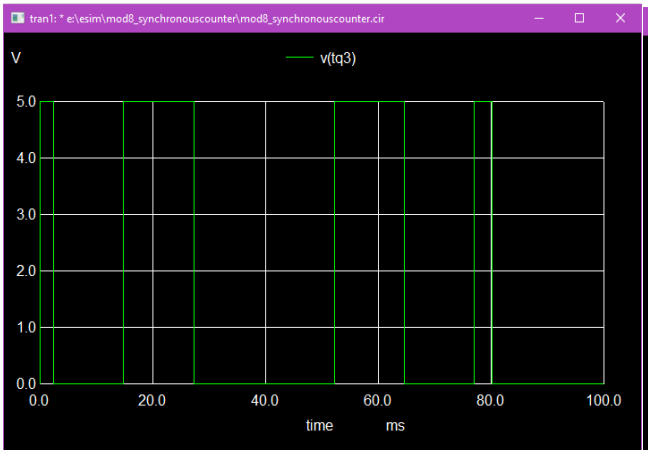


Fig.15: Output signal of T-FF 3 [Q_3]

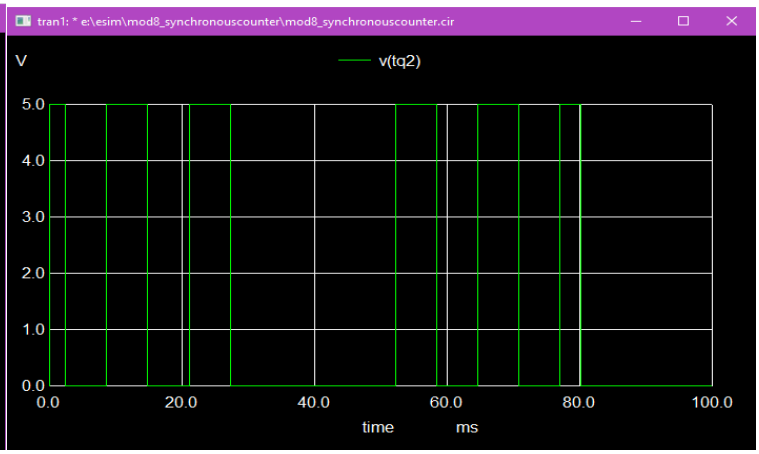


Fig.14: Output Signal of T-FF 2 [Q_2]

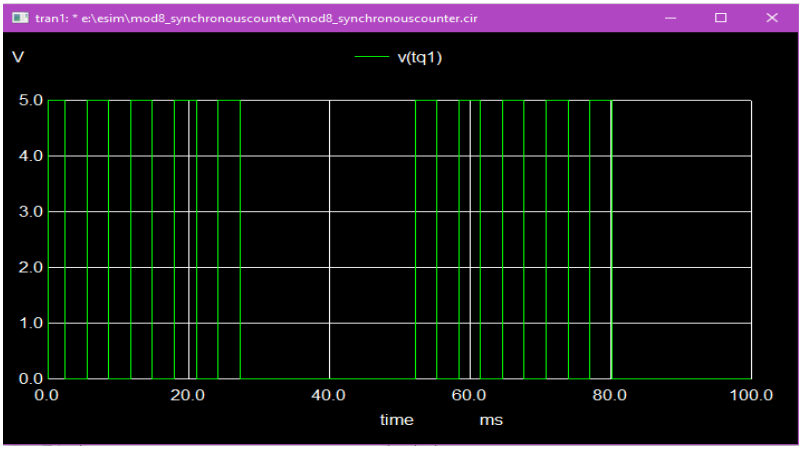


Fig.15: Output Signal of T-FF 1 [Q_1]

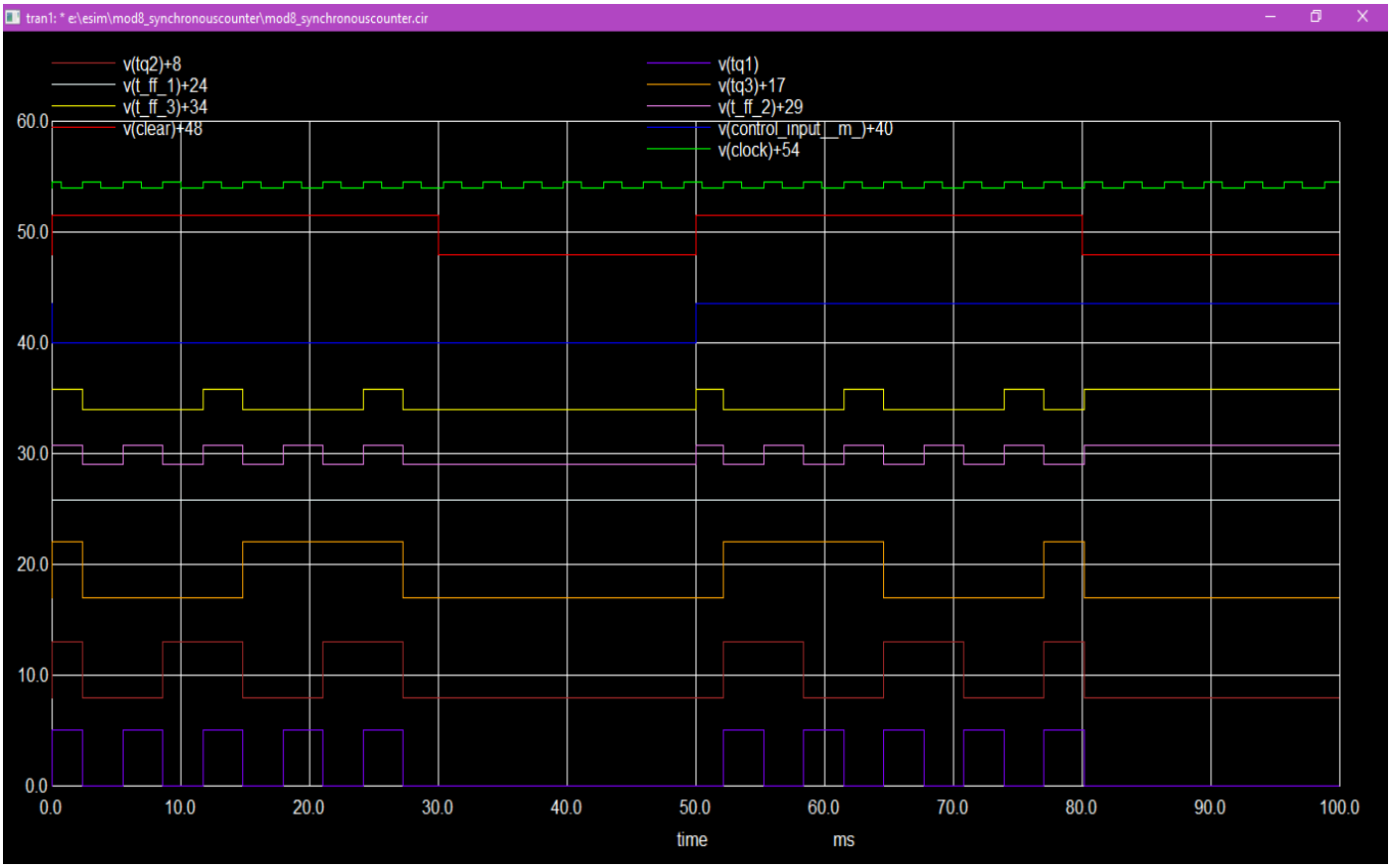


Fig.16: Combined Output Signal of MOD 8 Up/Down Synchronous Counter

Conclusion:-

CMOS gates are used in the combinational circuit's design to minimise power consumption and output delay. The eSim tool is used to design the circuits utilising 130nm technology.

References:-

[1]
https://www.researchgate.net/publication/312195862_Rapid_low_power_Synchronous_circuits_using_transmission_gates

[2]
Zhang, Tangbiao, and QingSheng Hu. "A high-speed and low-power up/down counter in 0.18-μm CMOS technology." 2012 International Conference on Wireless Communications and Signal Processing (WCSP). IEEE, 2012.

[3]
Github Link: https://github.com/Swagatika-Meher/Mod-8-Up-Down-Synchronous-Counter-using-130nm-CMOS-Technology-.git