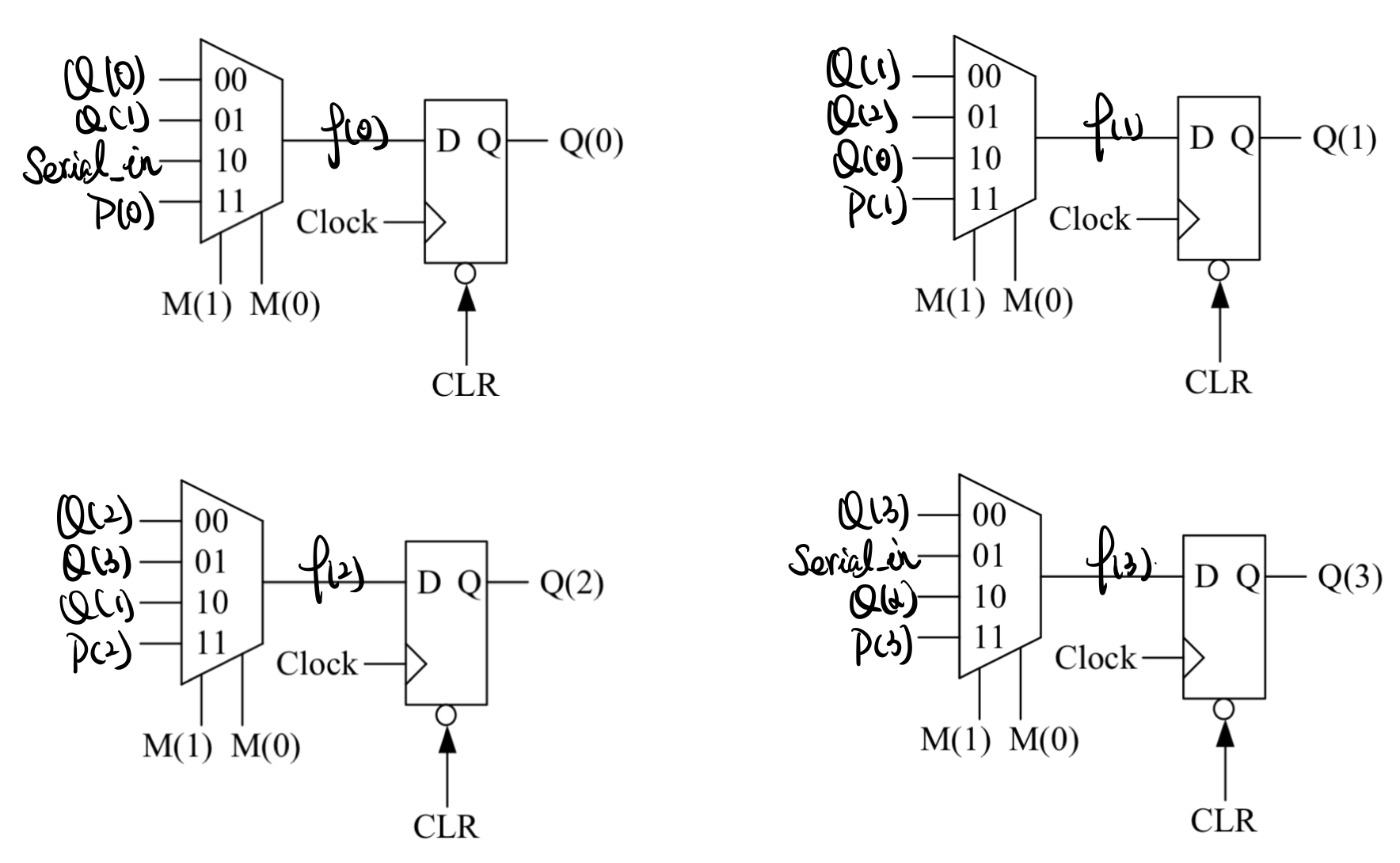
**EE 115B, Fall 2021**

**Student Name: 朱宇轩 2020531016 Lab 3**

1 Schematic (20 points.)



2 Code for Components (10 points. 5 points each.)

2.1 D flip flop

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Dflipflop is

Port ( D : in STD\_LOGIC;

clr : in STD\_LOGIC;

clock : in STD\_LOGIC;

Q : buffer STD\_LOGIC);

end Dflipflop;

architecture Behavioral of Dflipflop is

begin

process (clr ,clock )

begin

if clr = '0' then

Q <= '0';

elsif clock'Event and clock = '1' then

Q <= D;

end if ;

end process ;

end Behavioral;

2.2 mux4to1

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mux4to1 is

Port ( w0 : in STD\_LOGIC;

w1 : in STD\_LOGIC;

w2 : in STD\_LOGIC;

w3 : in STD\_LOGIC;

M : in STD\_LOGIC\_VECTOR (1 downto 0);

f : buffer STD\_LOGIC);

end mux4to1;

architecture Behavioral of mux4to1 is

begin

with M select

f <= w0 when "00",

w1 when "01",

w2 when "10",

w3 when others;

end Behavioral;

3 Code for Register (30 points.)

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity shiftreg is

Port ( P : IN std\_logic\_vector(3 downto 0);

Q : buffer std\_logic\_vector(3 downto 0);

clock : IN std\_logic;

clr : IN std\_logic;

serial\_in : IN std\_logic;

M : IN std\_logic\_vector(1 downto 0)

);

end shiftreg;

architecture Structure of shiftreg is

signal f : std\_logic\_vector (3 downto 0 );

component mux4to1

port ( w0 : in STD\_LOGIC;

w1 : in STD\_LOGIC;

w2 : in STD\_LOGIC;

w3 : in STD\_LOGIC;

M : in STD\_LOGIC\_VECTOR (1 downto 0);

f : buffer STD\_LOGIC);

end component ;

component Dflipflop

port ( D : in STD\_LOGIC;

clr : in STD\_LOGIC;

clock : in STD\_LOGIC;

Q : buffer STD\_LOGIC);

end component ;

begin

mux0: mux4to1 port map

(Q(0),Q(1),serial\_in ,P(0),M(1 downto 0),f(0));

mux1: mux4to1 port map

(Q(1),Q(2),Q(0),P(1),M(1 downto 0),f(1));

mux2: mux4to1 port map

(Q(2),Q(3),Q(1) ,P(2),M(1 downto 0),f(2));

mux3: mux4to1 port map

(Q(3),serial\_in ,Q(2),P(3),M(1 downto 0),f(3));

D0: Dflipflop port map

(f(0),clr ,clock ,Q(0));

D1: Dflipflop port map

(f(1),clr ,clock ,Q(1));

D2: Dflipflop port map

(f(2),clr ,clock ,Q(2));

D3: Dflipflop port map

(f(3),clr ,clock ,Q(3));

end Structure;

4 Code for Testbench (20 points.)

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

ENTITY shiftreg\_test IS

END shiftreg\_test;

ARCHITECTURE behavior OF shiftreg\_test IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT shiftreg

PORT(

P : IN std\_logic\_vector(3 downto 0);

Q : buffer std\_logic\_vector(3 downto 0);

clock : IN std\_logic;

clr : IN std\_logic;

serial\_in : IN std\_logic;

m : IN std\_logic\_vector(1 downto 0)

);

END COMPONENT;

--Inputs

signal P : std\_logic\_vector(3 downto 0) := (others => '0');

signal clock : std\_logic := '0';

signal clr : std\_logic := '0';

signal serial\_in : std\_logic := '0';

signal m : std\_logic\_vector(1 downto 0) := (others => '0');

--Outputs

signal Q : std\_logic\_vector(3 downto 0);

-- Clock period definitions

constant clock\_period : time := 20ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: shiftreg PORT MAP (

P => P,

Q => Q,

clock => clock,

clr => clr,

serial\_in => serial\_in,

m => m

);

-- Clock process definitions

clock\_process :process

begin

clock <= '0';

wait for clock\_period/2;

clock <= '1';

wait for clock\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

-- test CLR

clr<='0';

wait for 30ns;

-- test parallel load

clr<='1';

m<="11";

P<="0110";

wait for 40ns;

-- test right shift

m<="01";

serial\_in<='1';

wait for 80ns; -- shift 4 bits

-- test left shift

m<="10";

serial\_in <='0';

wait for 80ns; -- shift 4 bits

-- test no change

m<="00";

wait for 10ns;

serial\_in <='1';

P<="1111";

wait for 80ns;

wait;

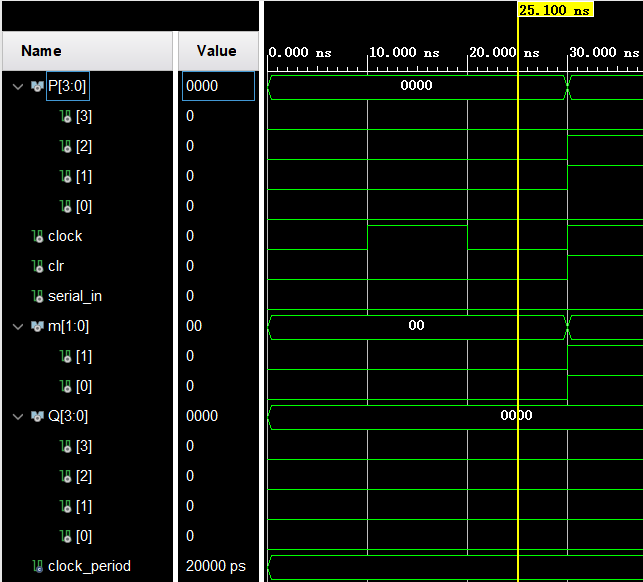
end process;

END;

5 Timing Diagram (20 points.)

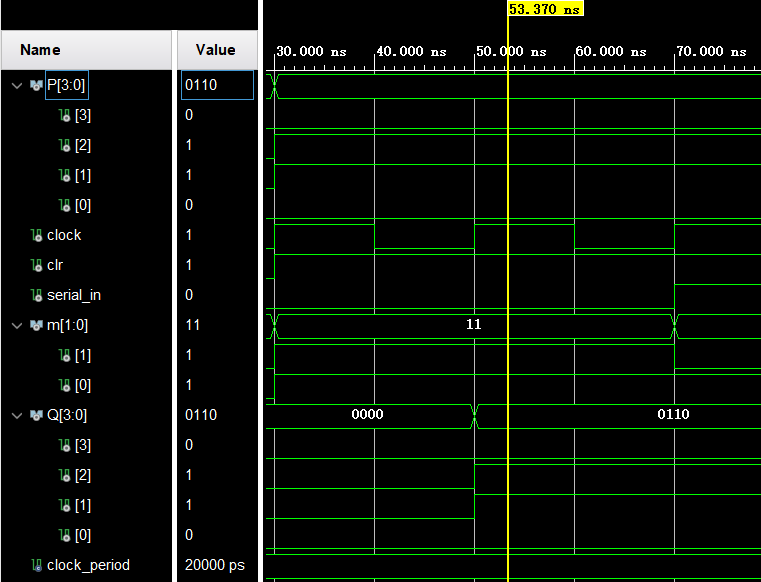
0-30ns: test CLR

All the outputs (Q(3:0)) maintains the value of 0 when clr=0.



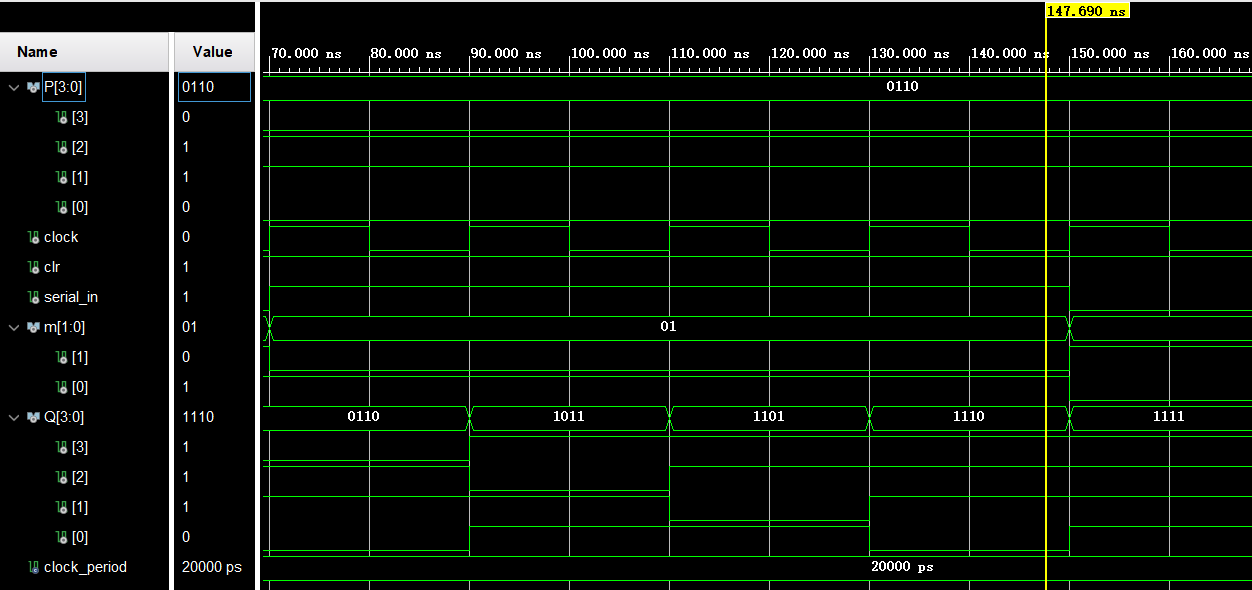
30-70ns: test parallel load

After the inputs have been set to the expected value, which is P(3:0)=”0110”, and waiting for a rising edge of the clock, the outputs turn into Q(3:0)=P(3:0)=”0110”, which verify the function of parallel load.



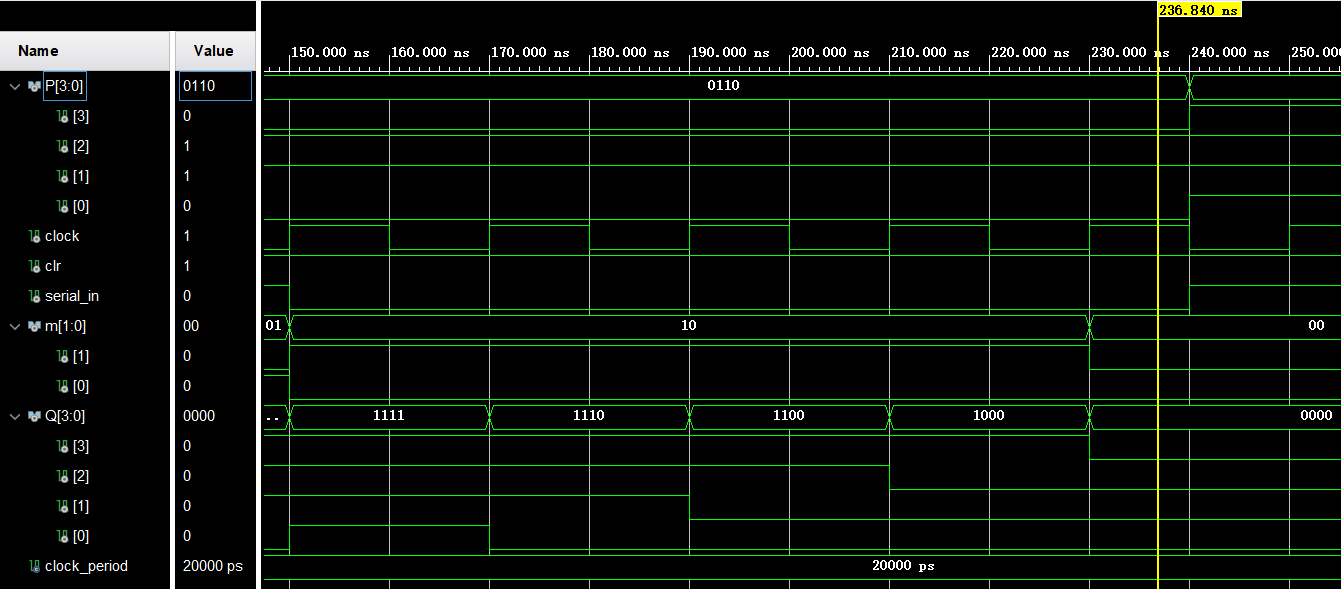
70-150ns: test right shift

Each rising edge of the clock makes all the data shifts to its right bit. And for Q(3), it uses the value of serial\_in, which always equal to 1. Moreover, the last rising edge happens at 150ns, thus the 4th shift result is shown in 150-170ns. Then from the simulation, we can see that the value of Q(3:0) changes as we expected, thus we verify the function of right shift.



150-230ns: test left shift

Each rising edge of the clock makes all the data shifts to its left bit. And for Q(0), it uses the value of serial\_in, which always equal to 0. Moreover, the last rising edge happens at230ns, thus the 4th shift result is shown in 230-250ns. Then from the simulation, we can see that the value of Q(3:0) changes as we expected, thus we verify the function of left shift.



230-∞ns: test no change

After setting M(1:0)=”00”, we change the value of P(3:0) and serial\_in for example, and we can see the value of Q(3:0) hold it original value at 230ns, thus we verify the function of no change.

