به نام خدا

تمرین چهارم

محمدمهدی آقاجانی

۹۳۳۱۰۵۶

استاد : دکتر صاحب الزمانی

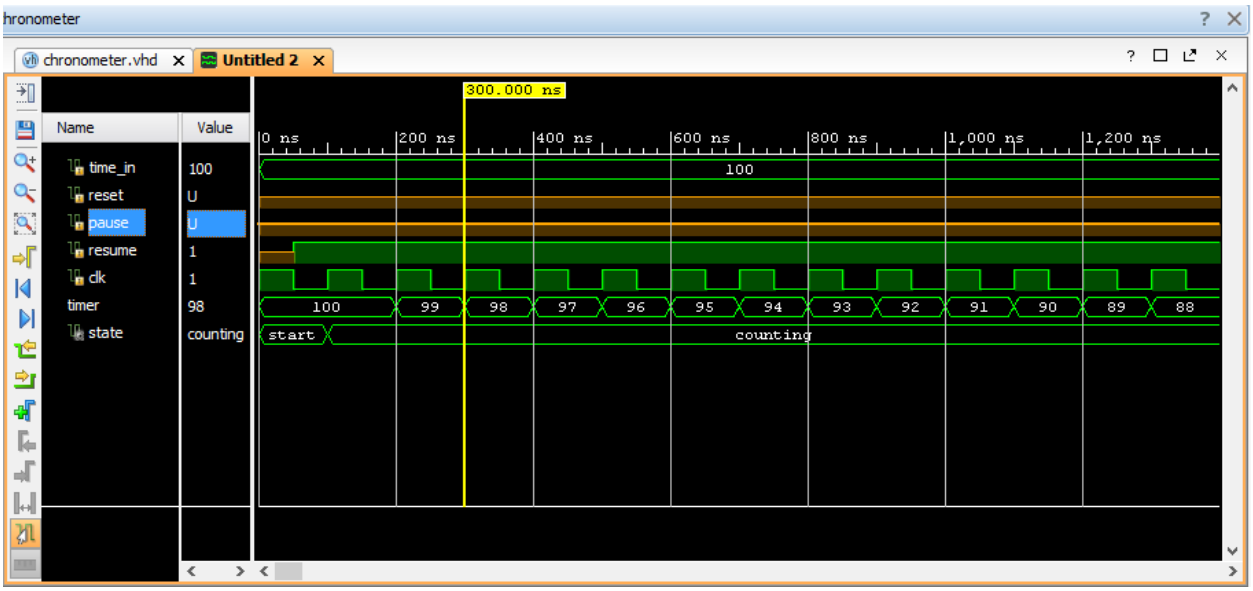
سوال پنجم

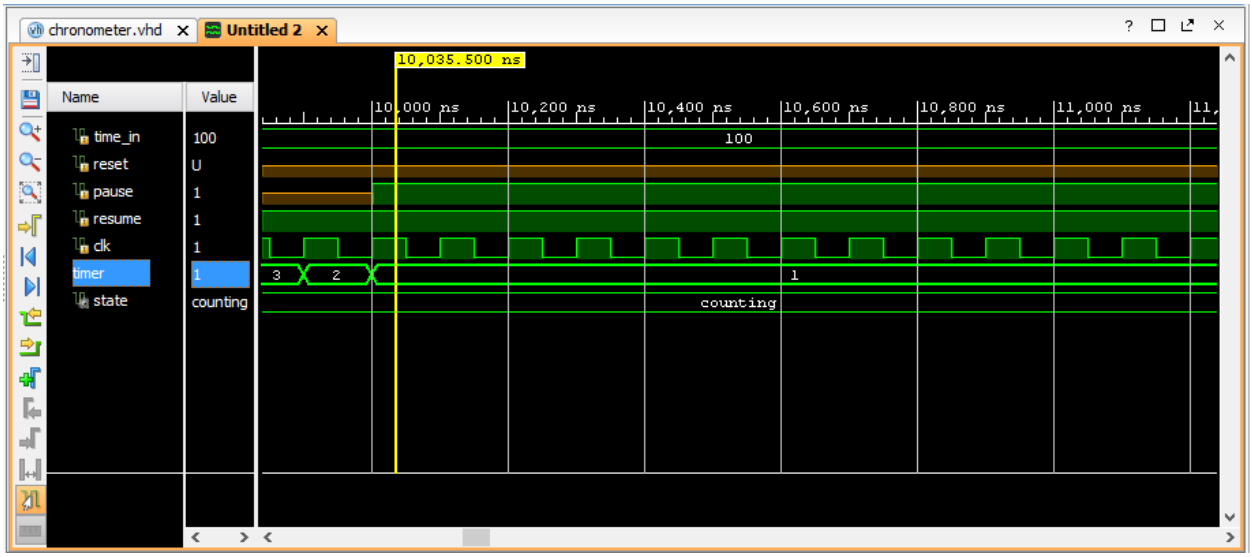
کد این سوال به صورت زیر است :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity chronometer is Port (
time_in : in integer ;
reset : in std_logic;
pause : in std_logic;
resume : in std_logic;
clk : in std_logic;
timer : inout integer
);
end chronometer;

architecture Behavioral of chronometer is
type State_type IS (start , counting);
signal state : state_type ;
begin
process(clk)
begin
    if( reset = '1' )then
        state <= start ;
    end if;
    if( clk'event and clk = '1')then
        case state is
            when start =>
                timer <= time_in;
                if( resume = '1' )then
                    state <= counting ;
                end if;
            when counting =>
                if( reset = '1' )then
                    state <= start;
                end if;

                if( resume = '1') then
                    if( pause = '1' )then
```

```
                                timer <= timer;
                                else
                                timer <= timer-1;
                            end if;
                else
                    if( pause = '1' )then
                                timer <= timer;
                                else
                                timer <= timer-1;
                            end if;
                end if;
            when others =>
                state <= start;
        end case;
    end if;
end process;

end Behavioral;
```

شکل موج تولید شده به صورت زیر است :





سوال ششم

کد این ماژول به صورت زیر است :

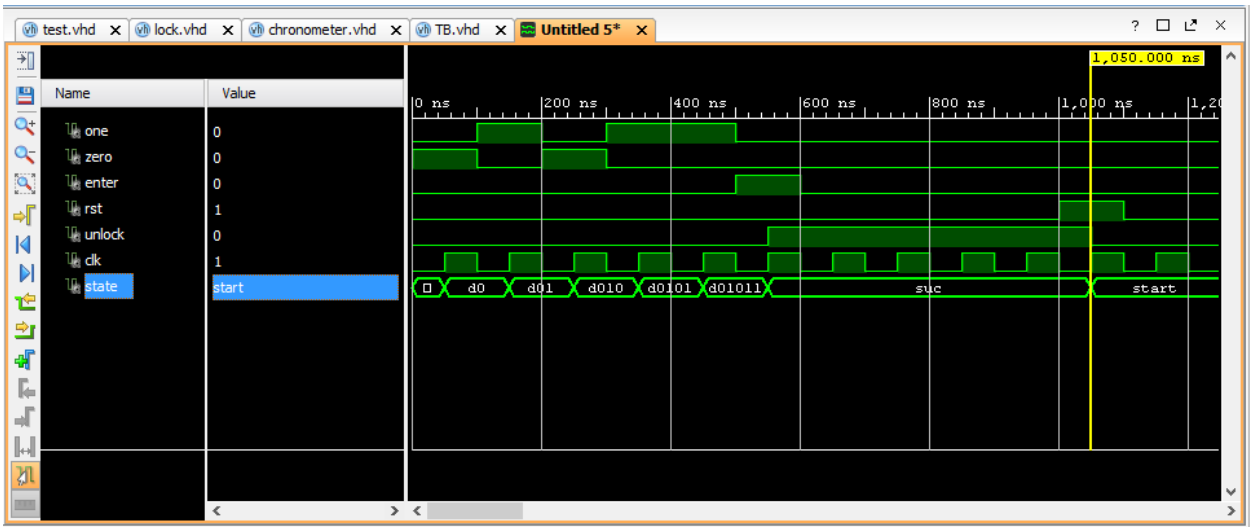
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lock is Port (
one , zero , enter , rst , clk :in std_logic;
unlock : out std_logic
 );
end lock;

architecture Behavioral of lock is
type state_type is ( start , d0 , d01 , d010 , d0101, d01011 , suc , fail
);
signal state : state_type;

attribute fsm_encoding : string;
attribute fsm_encoding of STATE : signal is "sequential";

begin
process(clk)
begin
```

```
if( clk'event and clk ='1')then
    if( rst = '1')then
        state <= start;
    end if;
    case state is
        when start =>
            if( rst = '1' or enter = '1' )then
                state <= start;
            elsif one = '1' then
                state <= fail ;
            elsif zero = '1' then
                state <= d0;
            else
                state <= start;
            end if;
        when d0 =>
            if( rst = '1' or enter = '1' ) then
                state <= start;
            elsif one = '1' then
                state <= d01;
            elsif zero = '1' then
                state <= fail;
            else
                state <= d0;
            end if;
        when d01 =>
            if( rst = '1' or enter = '1' ) then
                state <= start;
            elsif one = '1' then
                state <= fail;
            elsif zero = '1' then
                state <= d010;
            else
                state <= d01;
            end if;
        when d010 =>
            if( rst = '1' or enter = '1' ) then
                state <= start;
            elsif one = '1' then
                state <= d0101;
            elsif zero = '1' then
                state <= fail;
            else
```

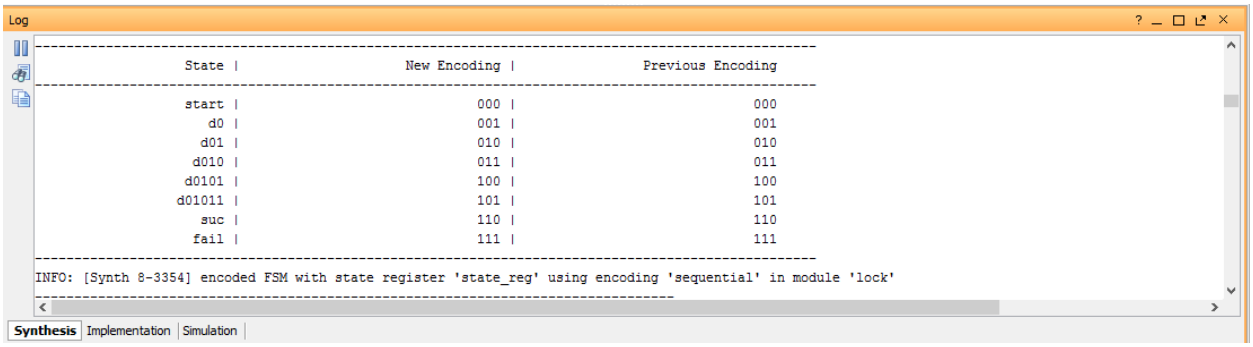
```
                state <= d010;
            end if;
        when d0101 =>
            if( rst = '1' or enter = '1' ) then
                state <= start;
            elsif one = '1' then
                state <= d01011;
            elsif zero = '1' then
                state <= fail;
            else
                state <= d0101;
            end if;
        when d01011 =>
            if( one = '1' or zero = '1') then
                state <= fail;
            elsif rst = '1' then
                state <= start;
            elsif enter = '1' then
                state <= suc;
            else
                state <= d01011;
            end if;
        when suc =>
            if( rst = '1' or enter = '1') then
                state <= start;
            elsif zero = '1' or one = '1' then
                state <= fail;
            else
                state <= suc;
            end if;
        when fail =>
            if( rst = '1' or enter = '1' ) then
                state <= start;
            elsif one = '1' or zero = '1' then
                state <= fail;
            else
                state <= fail;
            end if;
    end case;
  end if;
end process;

process( state )
```

```
begin
if( state = suc ) then
    unlock <= '1';
else
    unlock <= '0';
end if;
end process;

end Behavioral;
```

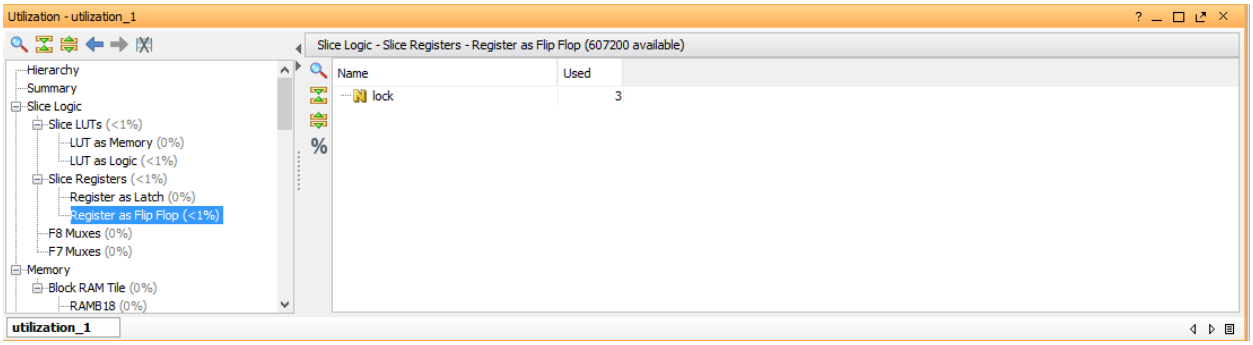
شکل موج آن به صورت زیر است :



در این طراحی از روش sequential استفاده شد زیرا تعداد حالت ها دقیقا برابر ۸ بود همچنین سیستم نیز از همین حالت استفاده کرده که گزارش آن در زیر موجود می باشد :



همچنین تعداد لچ ها برابر با صفر و تعداد فلیپ فلاپ ها ۳ عدد می باشد که آن هم در گزارش زیر موجود است :



و اما در مورد قسمت آخر هم کد آن به صورت زیر می باشد :

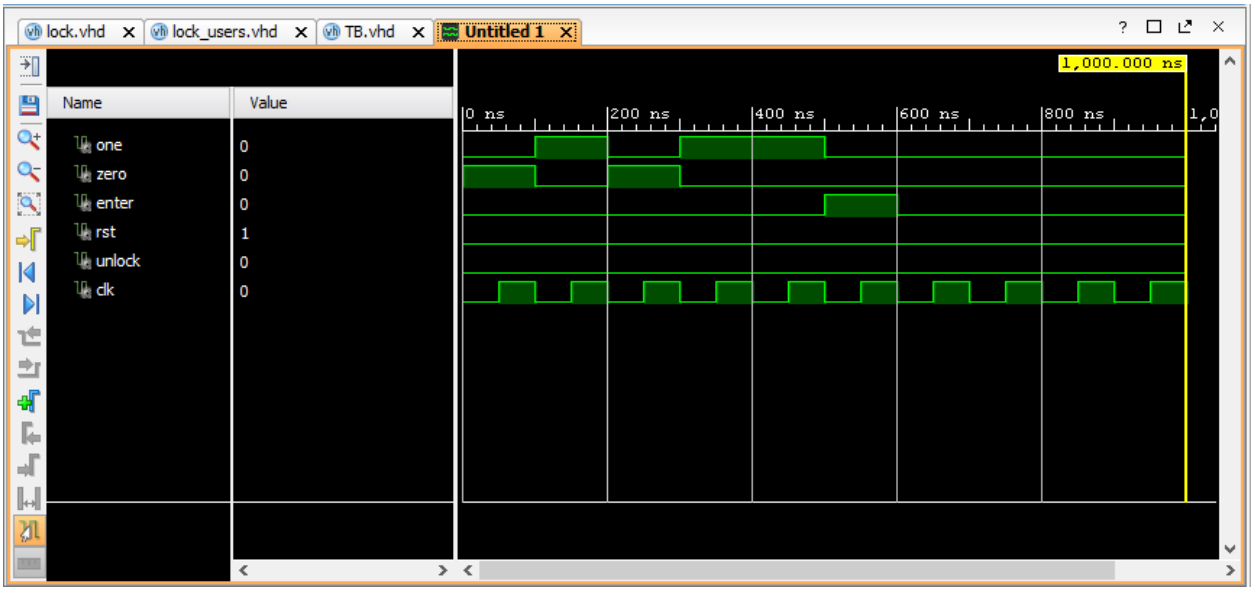
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity lock_users is Port (
one , zero , enter , rst , clk :in std_logic;
unlock : out std_logic
 );
end lock_users;

architecture Behavioral of lock_users is
type state_type is ( start , get_user_id , get_pass , get_new_pass ,
insert_id, insert_pass , suc , fail );
signal state : state_type;


attribute fsm_encoding : string;
attribute fsm_encoding of STATE : signal is "sequential";

begin
process(clk)

variable user_id1 : std_logic_vector( 1 downto 0 ) := "01";
variable user_id2 : std_logic_vector( 1 downto 0 ) := "10";
variable user_id3 : std_logic_vector( 1 downto 0 ) := "11";
```

```
variable temp_id : std_logic_vector( 0 to 1 );
variable index_id : integer range 0 to 1 := 0;

variable pass1,pass2,pass3 : std_logic_vector( 7 downto 0 );
variable temp_pass : std_logic_vector( 0 to 7 );
variable temp_new_pass : std_logic_vector( 0 to 7 );
variable index_pass : integer range 0 to 7 := 0;
variable index_new_pass : integer range 0 to 7 := 0;
begin
    if( clk'event and clk ='1')then
        if( rst = '1')then
            state <= start;
        end if;
        case state is
            when start =>
                if( one = '1' or zero = '1' )then
                    state <= insert_id;
                elsif rst = '1' then
                    state <= start ;
                elsif enter = '1' then
                    state <= get_user_id;
                else
                    state <= start;
                end if;
            when get_user_id =>
                if( zero = '1' or one = '1' ) then
                    if( zero = '1')then
                        temp_id(index_id) := '0';
                    else
                        temp_id(index_id) := '1';
                    end if;
                    index_id := (index_id + 1) mod 2;
                    state <= get_user_id;
                elsif enter = '1' then
                    state <= get_pass;
                elsif rst = '1' then
                    state <= start;
                else
                    state <= get_user_id;
                end if;
            when get_pass =>
                if( one = '1' or zero = '1' ) then
                    if( zero = '1')then
```

```
            temp_pass(index_pass) := '0';
        else
            temp_pass(index_pass) := '1';
        end if;
        index_pass := (index_pass + 1) mod 8;
        state <= get_pass;
    elsif enter = '1' then
        state <= get_new_pass;
    elsif rst = '1' then
        state <= start;
    else
        state <= get_pass;
    end if;
when get_new_pass =>
    if( one = '1' or zero = '1' ) then
        if( zero = '1' )then
            temp_new_pass(index_new_pass) := '0';
        else
            temp_new_pass(index_new_pass) := '1';
        end if;
        index_new_pass := (index_new_pass + 1) mod 8;
        state <= get_new_pass;
    elsif enter = '1' then
        if( pass1 = temp_pass)then
            case temp_id is
                when "01" =>
                    pass1 := temp_new_pass;
                when "10" =>
                    pass2 := temp_new_pass;
                when "11" =>
                    pass3 := temp_new_pass;
                when others =>
            end case;
        end if;
        state <= start;
    elsif rst = '1' then
        state <= start;
    else
        state <= get_new_pass;
    end if;
when insert_id =>
    if( one = '1' or zero = '1' ) then
        if( zero = '1')then
```

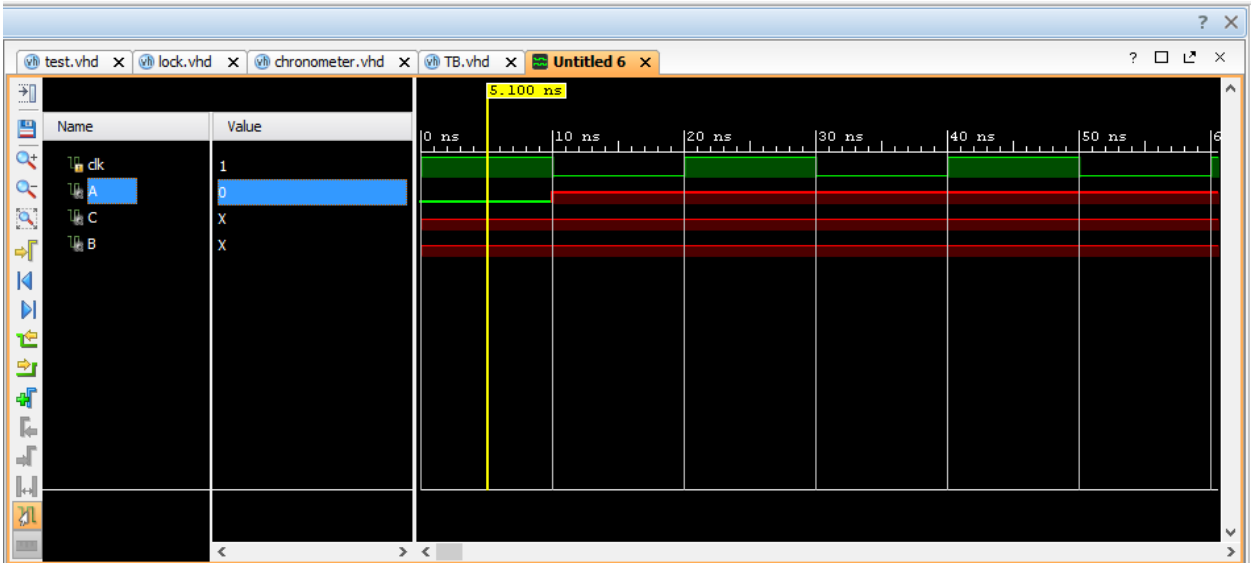
```
                temp_id(index_id) := '0';
            else
                temp_id(index_id) := '1';
            end if;
            index_id := (index_id + 1) mod 2;
            state <= insert_id;
        elsif enter = '1' then
            state <= insert_pass;
        elsif rst = '1' then
            state <= start;
        else
            state <= insert_id;
        end if;
    when insert_pass =>
        if( one = '1' or zero = '1') then
            if( zero = '1')then
                temp_pass(index_pass) := '0';
            else
                temp_pass(index_pass) := '1';
            end if;
            index_pass := (index_pass + 1) mod 8;
            state <= insert_pass;
        elsif rst = '1' then
            state <= start;
        elsif enter = '1' then
            case temp_id is
            when "01" =>
                if( pass1 = temp_pass)then
                    state <= suc;
                else
                    state <= fail;
                end if;
            when "10" =>
                if( pass2 = temp_pass)then
                    state <= suc;
                else
                    state <= fail;
                end if;
            when "11" =>
                if( pass3 = temp_pass)then
                    state <= suc;
                else
                    state <= fail;
```

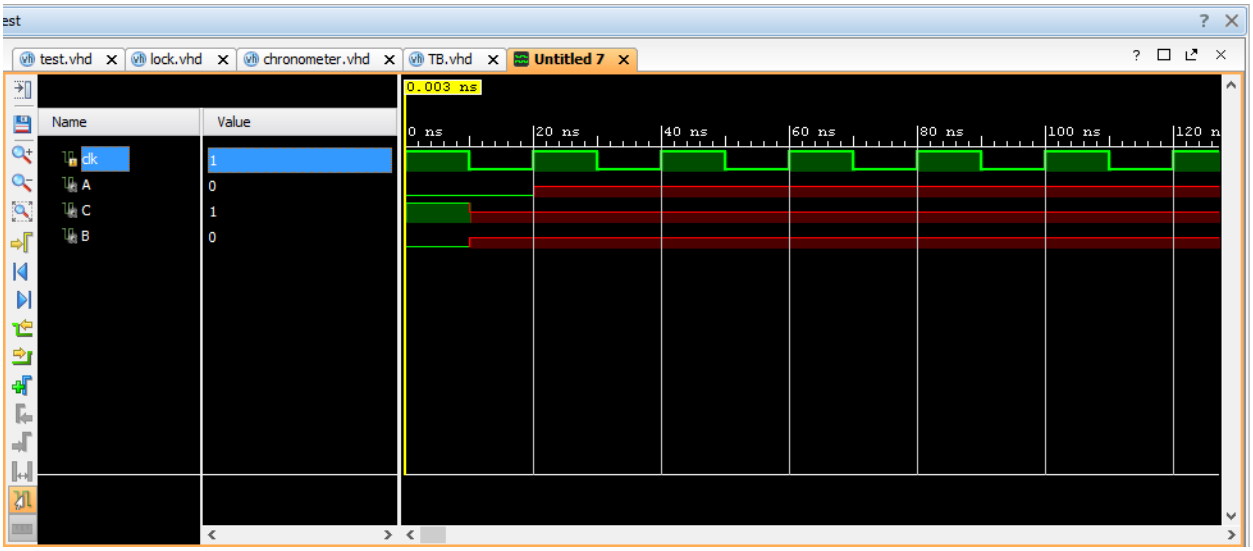
```
                    end if;
                when others =>
                end case;
            else
                state <= insert_pass;
            end if;
        when suc =>
            if( rst = '1' or enter = '1') then
                state <= start;
            elsif zero = '1' or one = '1' then
                state <= fail;
            else
                state <= suc;
            end if;
        when fail =>
            if( rst = '1' or enter = '1' ) then
                state <= start;
            elsif one = '1' or zero = '1' then
                state <= fail;
            else
                state <= fail;
            end if;
    end case;
end if;
end process;

process( state )
begin
if( state = suc ) then
    unlock <= '1';
else
    unlock <= '0';
end if;
end process;

end Behavioral;
```

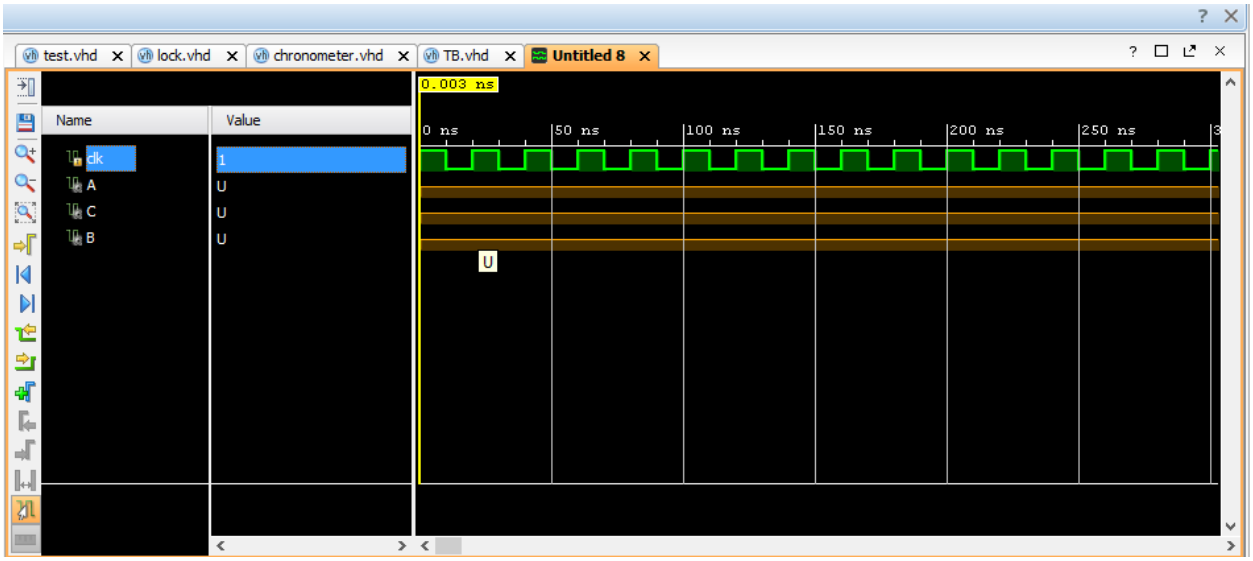


سوال هفتم

در این سوال برای سیگنال های B , C دو درایور در نظر گرفته شده است که همین امر باعث می شود مقدار آن ها در شرایطی تبدیل به X بشود . (البته کد داده شده اندکی مشکل دارد و باید سیگنال ها حتما مقدار اولیه بگیرند و الا ابزار vivado مقدار آن ها را برابر با U در نظر میگیرد. در تصاویر زیر انواع مقدار دهی اولیه انجام شده است و در آخری مقدار دهی اولیه نشده)







سوال هشتم

کد رمز کننده به صورت زیر است :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

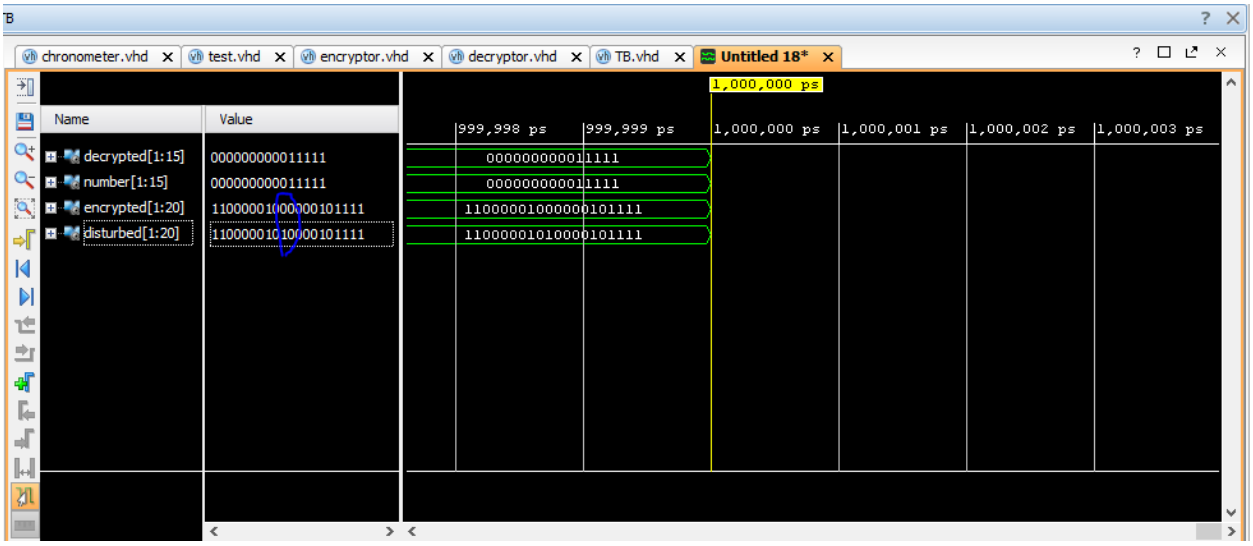
```
entity encryptor is Port (
input : in std_logic_vector( 1 to 15 );
output : out std_logic_vector( 1 to 20 )
);
end encryptor;

architecture Behavioral of encryptor is
signal p1 , p2 , p4 , p8 , p16 : std_logic;
begin
p1 <= input(1) xor input(2) xor input(4) xor input(5) xor input(7) xor
         input(9) xor input(11) xor input(12) xor input(14);
p2 <= input(1) xor input(3) xor input(4) xor input(6) xor input(7) xor
                  input(10) xor input(11) xor input(13) xor input(14);
p4 <= input(2) xor input(3) xor input(4) xor input(8) xor input(9) xor
                            input(10) xor input(11) xor input(15);
p8 <= input(5) xor input(6) xor input(7) xor input(8) xor input(9) xor
                                    input(10) xor input(11);
p16 <= input(12) xor input(13) xor input(14) xor input(15);

output <= p1 & p2 & input(1) & p4 & input( 2 to 4) & p8 & input( 5 to 11 )
& p16 & input( 12 to 15 );
end Behavioral;
```

کد رمز گشا به صورت زیر است :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

USE ieee.numeric_std.ALL;

entity decryptor is Port (
input : in std_logic_vector( 1 to 20 );
output : out std_logic_vector( 1 to 15)
);
end decryptor;

architecture Behavioral of decryptor is
signal numsig : integer ;
begin

process( input )
variable p1,p2,p4,p8,p16,tempsig : std_logic ;
variable num : integer range  1 to 20 ;
```

```
variable temp : std_logic_vector( 1 to 5 );
variable temp_input : std_logic_vector( 1 to 20 );
begin
p1 := input(1) xor input(3) xor input(5) xor input(7) xor input(9) xor
         input(11) xor input(13) xor input(15) xor input(17) xor
input(19);
p2 := input(2) xor input(3) xor input(7) xor input(6) xor input(10) xor
                  input(11) xor input(14) xor input(15) xor input(18) xor
input(19);
p4 := input(4) xor input(5) xor input(6) xor input(7) xor input(12) xor
                           input(13) xor input(14) xor input(15) xor
input(20);
p8 := input(8) xor input(9) xor input(10) xor input(11) xor input(12) xor
                                    input(13) xor input(14) xor input(15);
p16 := input(16) xor input(17) xor input(18) xor input(19) xor input(20);

if( p1 = '0' and p2 = '0' and p4 = '0' and p8 = '0' and p16 = '0' )then
output <= input(3)& input( 5 to 7 ) & input(9 to 15) & input(17 to 20);
else
temp := p16&p8&p4&p2&p1;
num :=  to_integer(unsigned(temp));
numsig <= num;
if( num > 0 and num < 16 ) then
temp_input := input;
tempsig := not temp_input(num);
temp_input(num) := tempsig;
output <= temp_input(3) & temp_input(5 to 7) & temp_input(9 to 15) &
temp_input(17 to 20);
end if;
end if;
end process;
end Behavioral;
```

شکل موج ها به صورت زیر است :



کد تست بنچ آن به صورت زیر نوشته شده است :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TB is
--  Port ( );
end TB;

architecture encryptor of TB is
component encryptor is Port (
input : in std_logic_vector( 1 to 15 );
output : out std_logic_vector( 1 to 20 )
);
end component;

component decryptor is Port (
input : in std_logic_vector( 1 to 20 );
output : out std_logic_vector( 1 to 15)
);
end component;

signal decrypted,number : std_logic_vector( 1 to 15);
signal encrypted , disturbed: std_logic_vector( 1 to 20 );
begin
number <= "000000000011111";
```

```
disturbed <= encrypted( 1 to 9 ) & not encrypted(10) & encrypted( 11 to
20);

encrypt: encryptor port map ( number , encrypted );
decrypt: decryptor port map ( disturbed , decrypted );

end encryptor;

architecture lock of TB is
component lock is Port (
one , zero , enter , rst , clk :in std_logic;
unlock : out std_logic
 );
end component;

signal one , zero , enter , rst , unlock: std_logic;
signal clk : std_logic := '0' ;
begin

MODULE: lock port map ( one , zero , enter , rst , clk , unlock );

one <= '0' ,
       '1' after 100ns,
       '0' after 200ns,
       '1' after 300ns,
       '0' after 500ns;
zero <= '1',
        '0' after 100ns,
        '1' after 200ns,
        '0' after 300ns ;
enter <= '0',
         '1' after 500ns,
         '0' after 600ns;
rst <= '0',
       '1' after 1000ns,
       '0' after 1100ns;

clk <= not clk after 50ns;

end lock;
```