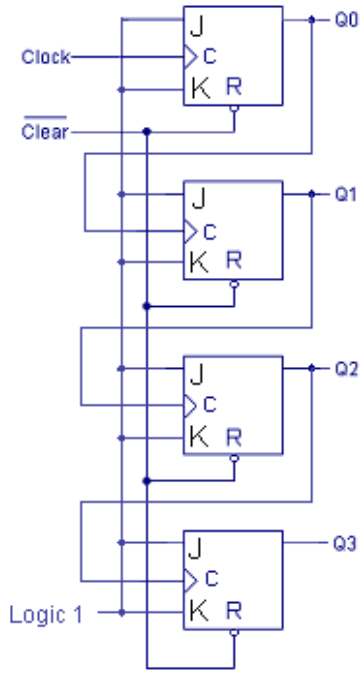
Лабораторийн ажил №5

Тоолуур

Зорилго: *Тоон системд өргөн хэрэглэгддэг төрөл бүрийн тоолуурын дизайныг хэрхэн VHDL код дээр хийж гүйцэтгэхийг судална.*

Асинхрон тоолуур

Оролтын импульсээр урьдчилан тодорхойлсон төлвүүдийн дарааллыг цувуулан гаргадаг регистрийг тоолуур гэдэг. Оролтын импульс нь клок импульс эсвэл өөр зарим үүсвэр байж болох ба тэдгээр нь хугацааны тогтмол интервалд эсвэл санамсаргүй интервалд ирдэг байж болно. Төлвүүд нь хоёртын тоон дарааллаар гардаг тоолуурыг хоёртын *(binary)* тоолуур гэдэг. *n*-бит хоёртын тоолуур нь *n* –бит флип-флопоос тогтох бөгөөд *0*-ээс 2^n*-1* хүртэл тоолно.



Зураг 1.

Тоолуурыг асинхрон (зарим тохиолдолд *ripple* ч гэдэг) ба синхрон гэж 2 янзаар ангилдаг. Асинхрон тоолуурт флип-флопын гаралтын шилжилт бусад флип-флопуудыг триггер хийх үүсвэр болдог. Өөрөөр хэлбэл бүх флип-флопууд ерөнхий клок импульсээр триггер хийгддэггүй харин бусад флип-флопуудын гаралтаар хийгддэг. 4 бит хоёртын асинхрон тоолуурыг зураг 1-д үзүүлэв. Бүх флип-флопын *J* ба *K* оролтууд нь байнгын логик 1–д холбогдсоноор тэдгээр нь *toggle* горим ажиллах буюу *C* оролтонд ирэх “буурах” фронтоор флип-флоп нь өөрийн утгаа инверслэнэ гэсэн үг. Флип-флоп бүрийн гаралт дараагийн флип-флопийн *C* оролтонд холбогдох байдлаар дараалжээ.

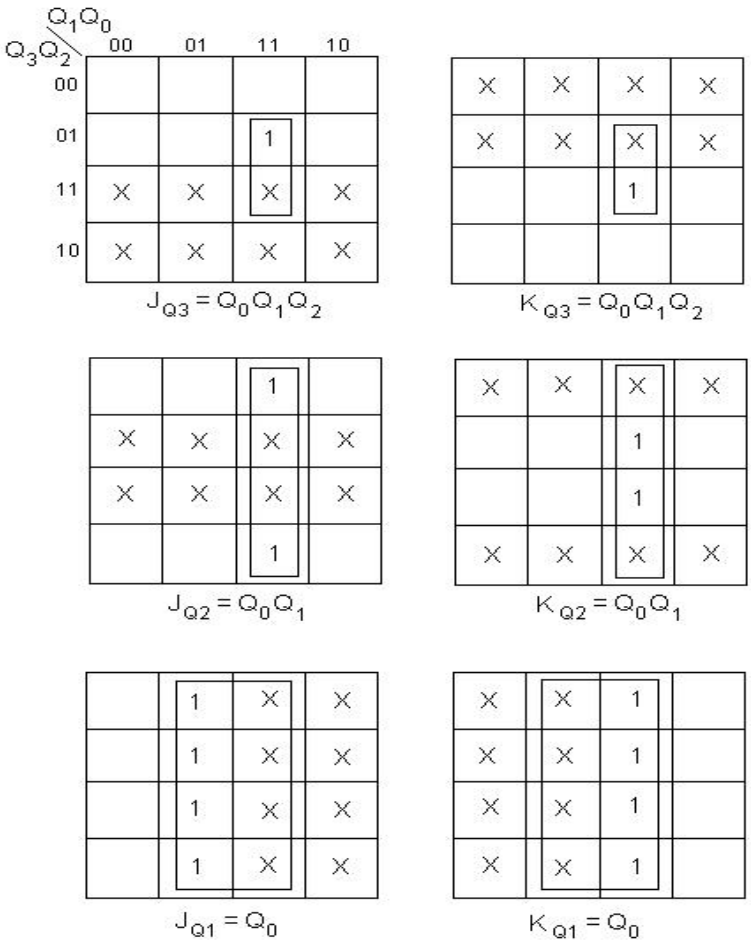
Асинхрон тоолуурын сайн тал нь түүний хялбар техникт байна. Гэвч асинхрон хэлхээ бөгөөд логик схем нэмсэн тохиолдолд тогтвортой биш, мөн хоцролтын хамааралтай байдаг. Мөн дамжиж хийгдэхэд шаардагдах хугацааны уртаас болж том хэмжээний асинхрон (*ripple*) тоолуур нь удаан хэлхээ болдог. Иймээс синхрон тоолуурыг өргөн хэрэглэдэг.

Хүснэгт 1.

Present state				Next state				Flip-flop inputs							
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J_{Q3}	K_{Q3}	J_{Q2}	K_{Q2}	J_{Q1}	K_{Q1}	J_{Q0}	K_{Q0}
0	0	0	0	0	0	0	1	0	×	0	×	0	×	1	×
0	0	0	1	0	0	1	0	0	×	0	×	1	×	×	1
0	0	1	0	0	0	1	1	0	×	0	×	×	0	1	×
0	0	1	1	0	1	0	0	0	×	1	×	×	1	×	1
0	1	0	0	0	1	0	1	0	×	×	0	0	×	1	×
0	1	0	1	0	1	1	0	0	×	×	0	1	×	×	1
0	1	1	0	0	1	1	1	0	×	×	0	×	0	1	×
0	1	1	1	1	0	0	0	1	×	×	1	×	1	×	1
1	0	0	0	1	0	0	1	×	0	0	×	0	×	1	×
1	0	0	1	1	0	1	0	×	0	0	×	1	×	×	1
1	0	1	0	1	0	1	1	×	0	0	×	×	0	1	×
1	0	1	1	1	1	0	0	×	0	1	×	×	1	×	1
1	1	0	0	1	1	0	1	×	0	×	0	0	×	1	×
1	1	0	1	1	1	1	0	×	0	×	0	1	×	×	1
1	1	1	0	1	1	1	1	×	0	×	0	×	0	1	×
1	1	1	1	0	0	0	0	×	1	×	1	×	1	×	1

Синхрон тоолуур

Синхрон тоолуур нь асинхрон буюу *ripple* тоолууpаас ялгаатай ба бүх флип-флопуудад нь клок импульс өгөгдсөн байдаг. Ингэснээр ерөнхий клок импульс бүх флип-флопуудыг нэгэн зэрэг триггер хийнэ. Синхрон тоолуурын дизайн процедур нь бусад синхрон цуваа схемийнхтэй адил байна. Тоолуур нь клок импульсээс өөр гадны оролтгүйгээр ажиллах боломжтой. Тоолуурын гаралт нь флип-флопуудын гаралт байх болно. Хоёртын тоолуурын төлвийн хүснэгт нь (*Хүснэгт 1*) одоогийн болон дараагийн төлвийг харуулсан багануудаас тогтоно. Мөн ямар флип-флопоор хийхээс хамаарч харгалзах флип-флопуудын оролтын багана байна. Манай жишээ хүснэгтэнд *JK* флип-флоп сонгогдсон нь харагдаж байна. *J* ба *K* оролтуудын утгыг *"excitation table"* –ээс олж тавьдагийг бид урьдах хичээлүүдээс мэдэх билээ. Дараа нь оролтын хялбарчлагдсан тэгшитгэлийг *зураг 2* -д үзүүлсэн байдлаар Карно карт хэрэглэн олно. Зурагт хамгийн бага битийн флип-флопийн 2 хүснэгтийг зураагүй байна. Эдгээр нь зөвхөн "1" болон *"don't care"* нөхцөлүүдээс тогтох тул J_{Q0} ба K_{Q0} нь хоёулаа "1"-тэй тэнцүү байна.



Зураг 2.

Ихэнх хэрэглээнд тоолуурын ажиллагааг удирдах үүднээс тоололтыг зөвшөөрөх (enable) оролт хэрэгтэй байдаг. Үүнийг *EN* гэж тэмдэглэе. Тэгвэл манай хоёртын тоолуурын флип-флопын оролтын тэгшитгэлүүд дараахь байдлаар илэрхийлэгдэх болно.

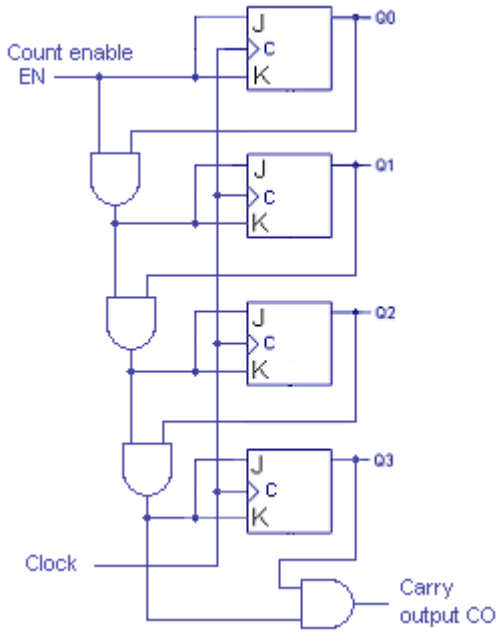
$$J_{Q0} = K_{Q0} = EN$$
$$J_{Q1} = K_{Q1} = Q_0 \cdot EN$$
$$J_{Q2} = K_{Q2} = Q_0 \cdot Q_1 \cdot EN$$
$$J_{Q3} = K_{Q3} = Q_0 \cdot Q_1 \cdot Q_3 \cdot EN$$

EN=0 үед бүх *J* болон *K* оролтууд 0 болох ба клок импульсууд ирж байсан ч бүх флип-флопууд төлвөө хадгалж үлдэнэ. *EN=1* үед эхний оролтын тэгшитгэл $J_{Q0} = K_{Q0} = 1$ болох ба бусад оролтын тэгшитгэлүүд Зураг 2-т бичигдсэн хэлбэртэй болно. Ингэснээр хоёртын синхрон тоолуурын хамгийн бага битийн флип-флоп нь клок импульсын шилжилт бүрд инверслэгдэнэ. Бусад флип-флопууд урьдах бүх бага битүүд нь1-тэй тэнцүү бол клок импульсийн шилжилт бүрд инверслэгдэнэ. Синхрон тоолуурын схемийг зураг 3-д үзүүлэв.

Бид түрүүчийн лабораторийн ажлаар хийж байсан *JK* флип-флопын кодоо ашиглан дээр өгүүлсэн асинхрон болон синхрон тоолуурын *VHDL* кодыг бичиж болно. *JK* флип-флопын кодыг дараахь байдлаар харуулав.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity JKFF is
port(clk, RST_n, J, K : in std_logic;
      Q, Qn           : out std_logic);
end JFKK;
```

```
architecture RTL of JKFF is
  signal FF :std_logic:='0';
begin
  process (clk, RST_n)
    variable JK : std_logic_vector(1 downto 0);
  begin
    if (RST_n = '0') then
      FF <= '0';
    elsif (clk'event and clk='1') then
      JK := J & K;
      case JK is
        when "01" => FF <='0';
        when "10" => FF <='1';
        when "11" => FF <= not FF;
        when others => FF <= FF;
      end case;
    end if;
  end process;
  Q <= FF after 1 ns;
  Qn <= not FF after 1 ns;
end RTL;
```



Зураг 3.

Дээрх кодыг ашиглан зураг 1-д үзүүлсэн асинхрон тоолууртай төстэй тоолуурыг *structural* хэлбэрээр бичвэл:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity AS_CNT is
port(RST_n, a_clock  : in std_logic;
      CNTR          : out std_logic_vector(3 downto 0));
end AS_CNT;

architecture RTL of AS_CNT is
  component JKFF
  port( clk, RST_n, J, K : in std_logic;
        Q, Qn            : out std_logic);
  signal FFQ : std_logic_vector(4 downto 0):="00000";
  signal FFQn : std_logic_vector(4 downto 0):="11111";
  signal VDD : std_logic:='1';
begin
  VDD <= '1'; FFQn(0) <= a_clock;
   jk0: for j in 1 to 4 generate
     uu0: JKFF port map (clk => FFQn(j-1), RST_n=>RST_n,
                J => VDD, K => VDD, Q => FFQ(j), Qn => FFQn(j));
   end generate;
   CNTR <= FFQ(4 downto 1);
end RTL;
```

болно. Энэхүү кодын ажиллагааг *ModelSim* симулятор дээр шалгаж үзээд асинхрон тоолуурын онцлогийг тэмдэглэн ав. Ямар тоонууд дээр түрүүчийн утгаас дараачийн жинхэнэ утгаа авах хүртэл хамгийн их хугацаа (*delay*) шаардагдаж байгааг анзаарна уу. Клокийн шилжилт хийгдсэнээс хойш жинхэнэ утгаа авах хүртэлх хугацаанд завсрын өөр утгууд гарч ирж байгаа учрыг тайлбарла.

Одоо *behavioral* хэлбэрээр бичигдсэн 8 бит синхрон тоолуурын кодыг авч үзье. Энд стандарт логик вектор *(std_logic_vector)* дээр арифметик үйлдэл хийгдэх тохиолдолд *std_logic_arith* ба *std_logic_unsigned* гэсэн хоёр шинэ санг *(library)* кодын эхэнд багтаах ёстой. *Unsigned library*-г хэрэглэсэн тул стандарт логик ба *integer* төрлүүдийн хооронд жиших үйлдэл зөвшөөрөгдөнө. *FF <= FF+1* гэсэн томъёололд *incrementer* буюу нэгээр нэмэгдүүлэгч үйлдэл орсон байна. *Incrementer*-ийн хэлхээг синтезлэхэд нэмэгчийнхээс (нэгийг нэмдэг *adder)* бага төхөөрөмж шаардагддаг. *VHDL* нь *"out"* сигналыг уншихыг зөвшөөрөхгүй тул *CNTR* гаралттай үргэлж адил байх дотоод сигнал *FF* –ийг хэрэглэсэн байна.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity S_CNT is
  port( clk, RST_n, EN : std_logic;
      CNTR    : out std_logic_vector(7 downto 0));
end S_CNT;
architecture RTL of S_CNT is
  signal FF : std_logic_vector(7 downto 0):="00000000";
begin
  process (clk, RST_n)
  begin
    if (RST_n='0') then
      FF <= (FF'range => '0');
    elsif (clk'event and clk = '1') then
      if (EN = '1') then
        FF <= FF+1;
      end if;
    end if;
  end process;
  CNTR <= FF;
end RTL;
```

Энэ кодоос харахад *EN* оролтын сигнал ‘1’ үед л клок сигналын фронтоор тоолуур утгаа нэгээр нэмэгдүүлэхээр байна.

Даалгавар

1. Дээрх хоёр жишээ кодыг *ModelSim* симулятор дээр шалгах *testbench* кодыг бичиж үр дүнг тэмдэглэж ав. Гаргаж авсан үр дүнгээсээ асинхрон болон синхрон тоолууруудын ажиллагааны ялгааг олж дүгнэлт хий.
2. Зураг 3-д үзүүлсэн *JK* флип-флоп дээр хийгдсэн тоолуурын *VHDL* кодыг *structural* хэлбэрээр бичиж өмнөх тоолууруудтай харьцуулж үз.
3. Дээшээ болон доошоо *(up-down)* тоолдог 8 бит *BCD* тоолуурын кодыг бичиж шалган үр дүнг харуул. Энд тоолуурт анхны утгыг өгөх параллель 8 бит оролт *(D0-D7)*, түүнийг тоолуурт бичих *load*, мөн буурах эсвэл өгсөх дарааллыг сонгох *direction* гэсэн нэмэлт оролтууд байх болно.
4. 8 бит Жонсоны тоолуурын кодыг бичиж шалган үр дүнг харуул.