Лабораторийн ажил №6

Төлвийн машин

Зорилго: *Цуваа тоон хэлхээний дизайнд хэрэглэгддэг төлвийн диаграм дээр үндэслэн төлвийн машин бүхий VHDL кодыг хэрхэн хийж гүйцэтгэхийг судална.*

Оршил

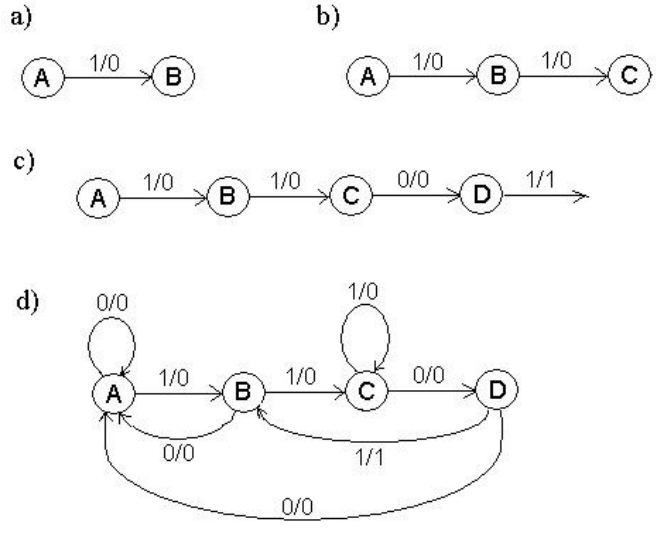
Цуваа *(sequential)* хэлхээний дизайн нь бодлогоо тодорхойлохоос эхлэх ба дараа нь Булийн функцүүдийн жагсаалт эсвэл логик диаграм хийж логик схемээ гаргаж авдаг. Комбинаци хэлхээнд үнэний хүснэгтээр бүгд бүрэн тодорхойлогддог бол цуваа хэлхээнд төлвийн хүснэгт шаардагддаг. Иймд цуваа схемийн дизайнд эхний шат нь төлвийн хүснэгт буюу эсвэл түүнтэй эквивалент төлвийн диаграм гаргаж авах юм.
Цуваа хэлхээний дизайн хийх процедур нь дараахь дарааллаар хийгдэнэ.

1. Бодлогын тодорхойлолтоос төлвийн диаграм эсвэл төлвийн хүснэгтийг гаргаж авна.
2. Төлвийн диаграм гаргаж авсан бол түүнээсээ төлвийн хүснэгтийг гаргаж авна.
3. Төлвүүддээ хоёртын код тодорхойлж өгнө.
4. Кодлогдсон төлвийн хүснэгт дэх дараагийн төлвийн оруулгаас флип-флопийн оролтын тэгшитгэлийг гаргаж авна.
5. Төлвийн хүснэгт дэх гаралтын оруулгаас гаралтын тэгшитгэлийг гаргаж авна.
6. Флип-флопийн оролтын тэгшитгэлүүд болон гаралтын тэгшитгэлүүдийг хялбарчилна.
7. Флип-флопийн оролтын тэгшитгэлүүд болон гаралтын тэгшитгэлүүдэд тодорхойлсоноор флип-флоп ба комбинаци гэйтүүдтэй логик диаграмыг зурна.

Хэлхээний тодорхойлолт нь голдуу хэлхээний ажиллагааг үгээр дүрсэлсэн хэлбэртэй байдаг. Дизайн процедурын эхний шатаар бол төлвийн диаграм эсвэл төлвийн хүснэгтийг олохын тулд энэхүү дүрслэлийг хөрвүүлэх хэрэгтэй. Хэлхээний төлвийн диаграм хэрхэн байгуулахыг дараахь жишээгээр үзье.

Тодорхой битийн цуваа гарч ирэхийг таньдаг хэлхээ байна гэж үзье. Жишээлбэл 1101 гэсэн битийн цувааг таних хэрэгтэй байсан гэе. Энэхүү цуваа-танигч нь ***X*** гэсэн нэг оролттой, ***Z*** гэсэн нэг гаралттай байна. Мөн энэ нь хэлхээний төлвийг бүгд тэг байх анхны байдал оруулах флип-флопийн шууд *reset*-тэй байна. Хэлхээний оролтонд ирсэн урьдах 3 бит нь **110** ба одоогийн оролт нь **1** байвал ***Z***-ийг **1** болгох замаар ***X*** оролтонд **1101** гэсэн битийн цуваа ирсэнийг таних хэлхээ байна. Бусад тохиолдолд ***Z*** нь **0** байна.

Төлвийн диаграм гаргаж авах гол түлхүүр нь өнгөрсөн оролтуудын түүхийг санахад хэрэглэгдэх төлвүүдийг таних явдал юм.. Жишээ нь: **1101** цувааны хувьд, цувааны сүүлийн **1**-тэй хамт гарах гаралтын утга **1**-ийг гаргахын тулд хэлхээ оролтын урдахь утгууд **110** байсныг санаж байгаа төлөвт байх ёстой. Энэ зарчмыг санаж, танигдах цувааны эхний хэсэг нь гарч ирээгүй байх төлөвийг **A** гэж тодорхойлон төлвийн диаграмаа томьёолохоос эхэлнэ. Хэрэв оролтонд **1** ирвэл, манай цувааны эхний бит **1** байх ёстой тул энэ үйл явдал хадгалагдах ёстой ба клок импульсийн дараа хэлхээний төлөв **A** байж таарахгүй. Иймээс хоёр дахь төлөв **B** нь цувааны эхний бит ирсэнийг харуулахаар байна. **1101** цувааны сүүлчийн **1** нь биш тул гаралт нь **0** байна. **B** төлөвт байж байхад нь оролтонд **1** ирсэн бол оролтын мөрийн хоёр ширхэг **1** ирсэнийг



Зураг 1.

төлөөлөх өөр нэг төлөв хэрэгтэй. Тиймээс **C** төлөв ба түүнтэй холбоотой шилжилт нэмэгдэнэ. Цувааны дараагийн бит нь **0**. Тэгвэл **C** төлөвт байхад энэ **0** ирвэл хоёр **1**, түүний араас нэг **0** ирсэнийг төлөөлөх төлөв хэрэгтэй бөгөөд энэ үед гаралт нь **0** байх нэмэлт төлөв **D** –г нэмж өгнө. **D** төлөвт байхад оролтонд **1** ирсэнээр таних ёстой цуваа маань дуусах ба **D** төлвөөс оролтын утга **1**-ын шилжилт нь гаралтын утга **1**-тэй байна. Энэ бүхнийг төлвийн диаграмаар хэсэгчилсэн байдлаар болон бүрнээр нь төлөөлүүлэн харуулсаныг *зураг 1*-ээс үзэж болно.
Энэхүү төлвийн диаграм нь *Mealy* машины хэлбэрээр хийгдсэн байгаа нь харагдаж байна. Төлвийн диаграм-д харгалзах төлвийн хүснэгтийг дараахь байдлаар хийвэл:

Одоогийн төлөв	Дараагийн төлөв		Гаралт, Z	
	X=0	X=1	X=0	X=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

Нэгэнт төлвийн диаграмаа гаргаж авсанаас хойш дараагийн процедур бол сонгож авсан флип-флопын оролтуудад зориулсан баганыг хүснэгтэнд нэмж, тэндээсээ флип-флопын “excitation” хүснэгтийг ашиглан флип-флопын оролтын логик схемийг гаргаж авдаг. Үүнийг бид урьд үзсэн “тоон систем” курс хичээлээр сайн мэдэх билээ. Энэхүү лабораторийн хичээлээр бид төлвийн диаграмаа ашиглан хэрхэн түүний VHDL кодыг бичихийг үзнэ.

Төлвийн даграммыг *VHDL* код руу хөрвүүлэх нь

Дээр өгүүлсэн цуваа танигчид зориулсан *VHDL* кодыг хэрхэн бичиж болохыг авч үзье. Архитектурыг нь нэгэн зэрэг гүйцэтгэгдэх, хамтран эзэмшиж буй сигналын утгуудаар хоорондоо харилцах 3 ялгаатай процессоос тогтсон байхаар хийе. Энд шинэ төрөл *(type)* тодорхойлсон төрлийн зарлалтыг оруулж өгнө. Төрөл зарлах нь *std_logic* зэрэг хэрэглэгдэж буй төрлүүдтэй адилаар шинэ төрлийг тодорхойлохыг зөвшөөрч байна гэсэн үг. Төрөл зарлахдаа ***type*** гэсэн үгээр эхлэх ба дараа нь шинэ төрлийнхөө нэр, тэгээд ***is*** гэж бичээд хаалтанд дотор шинэ төрлийн сигналын утгуудын жагсаалтыг бичиж өгнө. Манай тохиолдолд жишээ нь:

type *state_type* ***is*** *(A, B, C, D);*

Шинэ төрлийн нэр нь *state_type* ба энэ тохиолдолд утгууд нь *зураг 1* дэх төлвийн нэрүүд байна. Нэгэнт ***type*** гэж зарлагдсан бол энэ нь *signal* ба *variable* –уудыг зарлахад хэрэглэгдэж болно. Доор үзүүлэх жишээ кодон дахь

signal *state, next_state* ***:*** *state_type;*

гэсэн мөрөнд *state* ба *next_state* нь *state_type* төрлийн сигнал гэдгийг зааж өгч байна. Иймд *state* ба *next_state* –ууд нь **A**,**B**,**C**, **D** гэсэн утгууд авч болох нь. Гурван процесс бүр нь тусдаа функц гүйцэтгэх боловч нийлээд цуваа танигчийг үүрэг гүйцэтгэхээр ажиллах болно. Процесс 1 нь төлөвийн хадгалах хэсгийг дүрсэлж байгаа ба өсөх фронтоор триггер хийгдэх флип-флопийн дүрслэл байна. Сигналиуд нь state_type төрлийн, мөн *RESET* –ээр хийгдэх төлөв нь **0** –ээс өөр **A** төлөв байгааг анзаарна уу. Процесс 2 нь дараагийн төлвийн функцийг харуулж байна. Ерөнхийдөө комбинацийн логикийг дүрслэхийн тулд оролтын өөрчлөлт бүрд процесс гүйцэтгэгдэж байна гэдгээс бүх оролтууд *sensitivity list* –д байх ёстой. Процесс 3 нь гаралтын функцийг дүрсэлсэн байна. Процесс 2 дахь төлвийн илэрхийлэлтэй адилаар *case* тодорхойлогч хэрэглэсэн байна. Дараагийн төлөвт төлвийн нэрийг өгөхийн оронд ***Z*** –д **0** эсвэл **1** утгыг өгсөн байна. Энэ жишээ нь гаралт нь хэлхээний оролтын функц байдаг *Mealy* –н төлвийн машин байна. Хэрэв гаралт нь зөвхөн төлвөөсөө хамаараллтай *Moore* –н төлвийн машин байсан бол ***X*** оролт *sensitivity list* –д байхгүй, мөн *case statement* дотор *if-then-else* бүтэц байхгүй байх байсан.

```
library ieee;
use ieee.std_logic_1164.all;
entity seq_rec is
  port(clk, reset, X: in std_logic;
       Z : out std_logic);
end seq_rec;
```

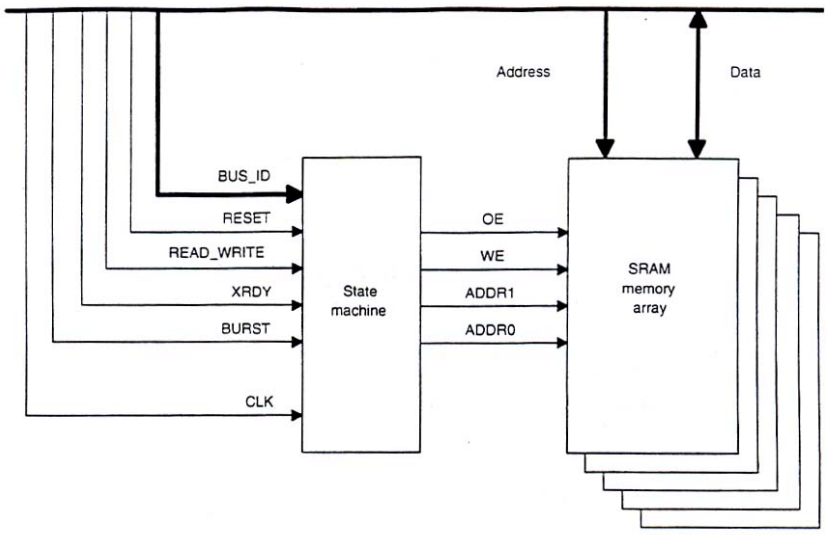
```
architecture process_3 of seq_rec is
   type state_type is (A,B,C,D);
   signal state, next_state : state_type;
begin
-- Процесс 1. асинхрон reset-тэй, өсөх фронтоор триггер хийгдэх төлөв хадгалах хэсэг
 state_register: process (clk, reset)
  begin
       if (reset='1') then
         state <= A;
       elsif (clk'event and clk = '1') then
         state <= next_state;
       end if;
   end process;

-- Процесс 2. next_state –ийг X оролт ба state –ийн функц байхаар гүйцэтгэнэ.
 next_state_func: process (X, state)
  begin
    case state is
      when A =>
        if X='1' then next_state <= B;
         else next_state <= A;
        end if;
      when B =>
        if X='1' then next_state <= C;
         else next_state <= A;
        end if;
      when C =>
        if X='1' then next_state <= C;
         else next_state <= D;
        end if;
      when D =>
        if X='1' then next_state <= B;
         else next_state <= A;
        end if;
     end case;
  end process;

-- Процесс 3. Гаралтыг X оролт ба state –ийн функц байхаар гүйцэтгэнэ.
output_func: process (X, state)
  Begin
    case state is
      when A =>  Z<='0';
      when B =>  Z<='0';
      when C =>  Z<='0';
      when D =>
        if X='1' then Z<='1';
        else Z<='0';
        end if;
    end case;
  end process;
end;
```

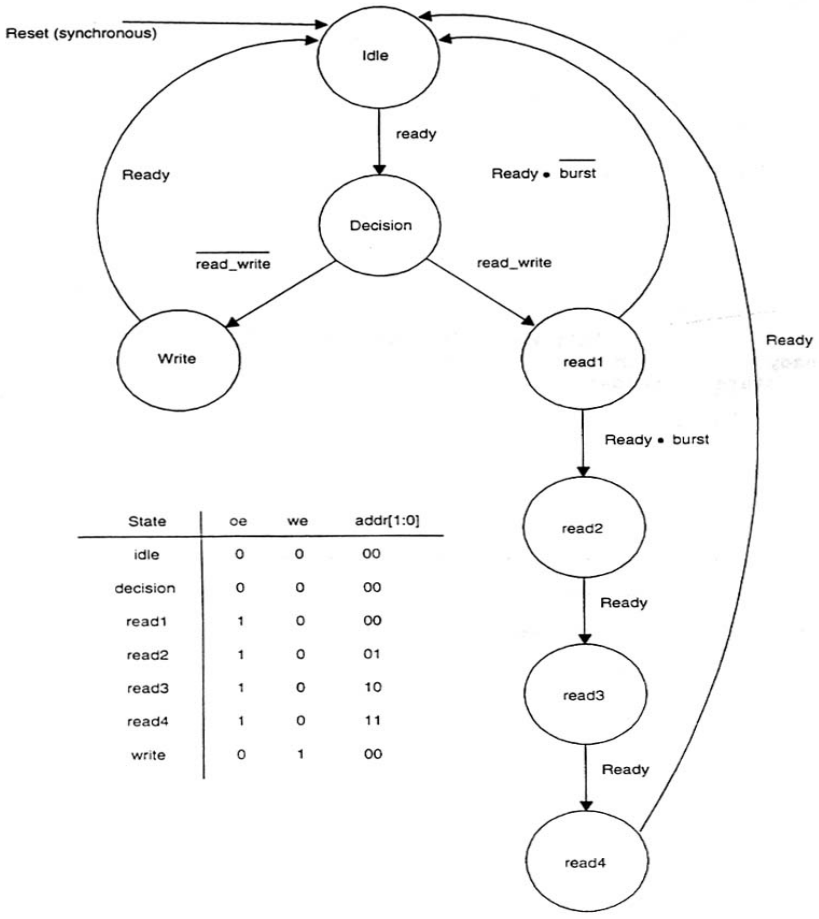
Даалгавар

1. Дээрх битийн цуваа танигчийн кодыг *ModelSim* симулятор програм дээр туршиж, үр дүнг тайланд хавсарга.
2. Дараахь зурагт санах ойн удирдлагын ерөнхий схем болон түүний төлвийн диаграмыг үзүүлсэн болно. Энэ нь төлвийн машины зарчим дээр ажиллах санах ойн удирдах хэсэг (контроллер) –ийг харуулсан байна. Төлвийн машин нь *reset, clk, ready, read_write, burst* гэсэн удирдлагын оролтуудаас гадна *BUS_ID* гэсэн өгөгдлийн оролттой байна. Энэ өгөгдлийн оролтын тодорхой утгаас (*мухайлбал 11110011*) хамаарч төлвийн машин маань идэвхжихээр байна. Мөн *OE* ба *WE* гаралтууд нь санах ойгоос өгөгдөл унших, түүн рүү бичихэд зориулагдсан удирдлагын гаралтууд бөгөөд дээр нь 2 бит *ADDR* гаралт нь санах ойн аль нэг банкийг сонгоход хэрэглэгдэхээр байна. Төлвийн

машины диаграммыг ашиглан түүний *VHDL* кодыг зохион, шалгаж үр дүнг харуул. Төлвийн машины маань код нь *case-when* бүтцээр хялбар хийгдэж болох ба загвар болгон кодын зарим хэсгийг бичиж үзүүлэв.

```
case present_state is
  when idle  => oe <='0'; we <= '0'; addr <= "00";
    if (bus_id) = "11110011"  then next_state <= decision;
    else next_state <= idle;
    end if;
  when decision => . .
    .
    .
  when read1 => . . .
    .
    .
  when read2 => . . .
    .
  when read3 => . . .
    .
  when read4 => . . .
    .
  when write => . . .
    .
    .
end case;
```





State	oe	we	addr[1:0]
idle	0	0	00
decision	0	0	00
read1	1	0	00
read2	1	0	01
read3	1	0	10
read4	1	0	11
write	0	1	00

Зураг 2.

Санах ойн удирдлагын төлвийн диаграм