Лабораторийн ажил №3

VHDL дэх санах элементийн төлөөлөл

Зорилго: *Тоон хэлхээний санах элементүүд болох latch, төрөл бүрийн flip-flop -ийн VHDL синтез кодыг хэрхэн бичихыг судална.*

Лач *(latch)*

Лач нь клок логик ‘1’ үед оролтон дахь өгөгдлийг гаралтанд гаргадаг. Харин клокийн ‘0’ байх үеийн турш лач нь гаралтан дээрх сүүлчийн утгаа хадгалж байдаг.
Лач-ийн синтез кодын жишээг харуулбал:

```
library IEEE;
use IEEE.std_logic_1164.ALL;

entity latch is
port ( data_in : IN std_logic;
       clock   : IN std_logic;
       data_out: OUT std_logic);
end latch;

architecture latch_arch of latch is
 begin
  process  (data_in, clock)
  begin
    if (clock='1') then
      data_out <= data_in;
    end if;
  end process;
end latch_arch;
```

VHDL синтез код эхлэн бичиж суралцагсад өөрсдийн дизайнд хүсээгүй лачууд синтезчлэлийн үед бий болсон байхыг олонтоо хардаг. Хэрэв дизайнер кодондоо хэд хэдэн чухал структурыг дутуу орхигдуулсан бол синтезийн *tool* тэдний хүсээгүй лач-уудыг байх юм байна гэж ойлгон автоматаар оруулчихдаг. Иймэрхүү “нуугдмал” лач-ууд дизайнер код бичих үедээ бүх гарч болох боломжийг бүрнээр нь тооцолгүй хийснээс гарч ирдэг. Жишээ нь бүх боломжит логик комбинацийг нь оруулаагүй *IF* оператор хэрэглэсэн үед нуугдмал лач үүсэх үндэс болдог. Үүнийг дараахь жишээнээс харж болно.

```
library IEEE;
use IEEE.std_logic_1164.ALL;

entity ifwlatch is
port (  input1  : IN std_logic;
        input2  : IN std_logic;
        input3  : IN std_logic;
        input4  : IN std_logic;
        selector: IN std_logic_vector(1 downto 0);
        output1 : OUT std_logic);
end ifwlatch;

architecture ifwl_arch of ifwlatch is
 signal internal : std_logic := '0';
 begin
  process  (input1, input2, input3, input4, selector)
  begin
    if (selector="00") then
      internal <= input1;
    elsif (selector="01") then
      internal <= input2;
```

```
    elsif (selector="10") then
      internal <= input3;
    end if;
  end process;
  output1 <= internal;
end ifwl_arch;
```

Энд *selector*-ийн бүх комбинацийг оруулаагүй байна. Иймээс энэ нь цаад утгаараа “лач” гарч ирэх үндэслэлийг агуулж байна. Учир нь *selector=”11”* үед *internal* сигнал буюу *output1* гаралт яаж тодорхойлогдохыг заагаагүйгээс синтезийн *tool* энэ тохиолдолд урьдахь утгаа хэрэглэх юм байна гэж үзэн түүнийг нь хадгалах лач автоматаар оруулж ирнэ. Лач гарч ирэхгүй байхаар *selector*-ийн бүх комбинацийг тодорхойлон энэ кодыг өөрчлөн бичвэл:

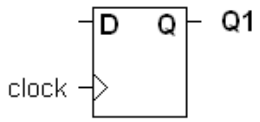
```
library IEEE;
use IEEE.std_logic_1164.ALL;

entity ifwol is
port (  input1  : IN std_logic;
        input2  : IN std_logic;
        input3  : IN std_logic;
        input4  : IN std_logic;
        selector: IN std_logic_vector(1 downto 0);
        output1 : OUT std_logic);
end ifwol;

architecture ifwol_arch of ifwol is
 signal internal : std_logic := '0';
 begin
  process  (input1, input2, input3, input4, selector)
  begin
    if (selector="00") then
      internal <= input1;
    elsif (selector="01") then
      internal <= input2;
    elsif (selector="10") then
      internal <= input3;
    elsif (selector="11") then
      internal <= input4;
    end if;
  end process;
  output1 <= internal;
end ifwol_arch;
```

болно. Ийм тохиолдол мөн *CASE statement*-ийг хэрэглэх үед гарч болдог.

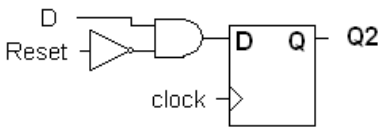
Флип-флоп

Флип-флоп нь лач-тай төстэй боловч клокийн шилжилтээр оролтыг лач хийдэг. Өөрөөр хэлбэл клокийн *low* –ээс *high* болох шилжилт (үүнийг *positive edge triggered* гэж нэрлэдэг) эсвэл *high*-аас *low* болох шилжилт (үүнийг *negative edge triggered* гэж нэрлэдэг) –ээр оролтын утгаас хамааран гаралтын төлөв нь тодорхойлогддог. Комбинацийн (хам) логик схемүүдээс ялгаатай нь флип-флоп зөвхөн *process* дотор байж синтезлэгддэг. VHDL -д клок сигналийн өөрчлөлтийг *clock’EVENT* –ээр илэрхийлдэг. Өөрөөр хэлбэл, клок ямар ч үед өөрчлөгдөхөд *clock’EVENT* үнэн (*true*) байна гэсэн үг. Клокийн өсөх фронт (*positive edge*) нь
clock’EVENT and clock=’1’ гэж сонгогдоно. Клокийн буурах фронт (*negative edge*) нь
clock’EVENT and clock=’0’ гэж сонгогдоно.
Хэрэв зөвхөн clock=’1’ гэж соливол түвшинээр төлвөө солидог лач болон синтезлэгдэнэ.
clock’EVENT and clock=’1’ -ийн оронд *rising_edge(clock)* гэж хэрэглэж болно. Энэ хоёр адил утгатай юм.
Дараахь жишээгээр клокийн өсөх фронтоор төлвөө солидог ***D*** флип-флопын янз бүрийн *reset* болон *enable* –тэй хувилбаруудыг харуулав.

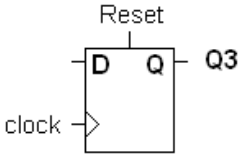
```
library IEEE;
use IEEE.std_logic_1164.ALL;

entity DFFs is
port (      D, Clock, Reset, Enable : IN std_logic;
      Q1, Q2, Q3, Q4          : OUT std_logic);
end DFFs;

architecture dff_behav of DFFs is
 begin
  process (clock)    -- өсөх фронтоор төлвөө солидог(positive edge triggered) D флип-флоп
  begin
    if (clock'EVENT AND clock='1') then
      Q1 <= D;
    end if;
  end process;
```



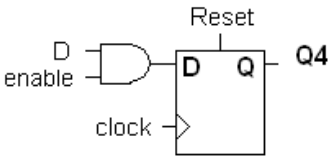

```
  process (clock)     -- Синхрон reset-тэй
  begin               -- өсөх фронтоор төлвөө солидог(positive edge triggered) D флип-флоп
if (clock'EVENT AND clock='1') then
      if Reset='1' then
        Q2 <= '0';
      else
        Q2 <= D;
      end if;
    end if;
  end process;
```




```
  process (clock, Reset)    -- Асинхрон reset- тэй
  begin                -- өсөх фронтоор төлвөө солидог(positive edge triggered) D флип-флоп
      if Reset='1' then
        Q3 <= '0';
      elsif (clock'EVENT AND clock='1') then
        Q3 <= D;
      end if;
  end process;
```




```
  process (clock, Reset) -- Асинхрон reset болон зөвшөөрөх сигнал (enable)-тай
  begin              --  өсөх фронтоор төлвөө солидог(positive edge triggered) D флип-флоп
      if Reset='1' then
        Q4 <= '0';
      elsif (clock'EVENT AND clock='1') then
        if Enable = '1' then
          Q4 <= D;
        end if;
      end if;
  end process;

end dff_behav;
```



Даалгавар

1. Дээрх лач болон D флип-флопийн кодуудыг ModelSim симулятор хэрэглэн шалгаж турш. Үүний тулд дээрх кодууддаа тохируулан testbench кодыг бичих эсвэл Modelsim симуляторын өөрийнх нь командуудыг хэрэглэн оролтын сигналуудын утгыг янз бүрээр өөрчлөн гаралтын waveform-ийг гарган авч болно. Өөрөөр хэлбэл оролтын өөрчлөлтүүдийг (stimuli) багтаасан modelsim –ийн командуудын макро файл үүсгэн түүнийгээ ажиллуулан шалгаж болно. Макро файл хэрэглэснээр олон тооны давтагдах гар ажиллагааг хөнгөвчлөх сайн талтай байдаг. Энд хамгийн их хэрэглэгдэх команд бол ***force*** юм. Үүний тусламжтай тухайн оролтын сигналыг хугацааны ямар үед ямар утгатай байхыг зааж өгдөг. Жишээ нь *data_in* сигналыг хугацааны 0 нс-ээс эхлэн ‘0’

утгатай байна, 105нс-ээс эхлэн 145нс хүртэл ‘1’ утгыг авна, 145нс-ээс цааш ‘0’ байна гэдгийг дараахь байдлаар зааж өгнө.

force -freeze data_in 0 0, 1 105ns, 0 145ns

Хэрэв 20нс-ийн үетэй *50% duty cycle* бүхий clock сигналыг

force -freeze -repeat 20ns clock 0 0, 1 10ns

гэж өгч болно. Гэхчлэн хэрэгтэй командуудаа нэг файлд бичиж хадгалаад түүнийгээ симуляторын ажлын цонхноос ***do*** командын тусламжтай гүйцэтгүүлнэ. Жишээ нь *D* флип-флопын кодыг шалгах макро командуудын файлыг *DFFs.m* нэртэй хадгалсан бол түүнийгээ

do *DFFs.m*

гэж бичин ажиллуулна. D флип-флопын кодыг шалгах симуляторын командуудыг дараахь байдлаар жишээ болгон харуулбал:

```
vsim work.dffs
view structure
view signals
add wave sim:/dffs/*
force -freeze -repeat 100ns D 0 0, 1 30ns
force -freeze -repeat 20ns clock 0 0, 1 10ns
force -freeze -repeat 105ns reset 1 10ns, 0 45ns, 1 105ns
force -freeze -repeat 110ns enable 0 0, 1 65ns
run 240ns
```

байна. Энд эхлээд симуляцид бэлтгэх командууд өгөгдсөн байна. Тухайлбал: *work* фолдер дахь *dffs* гэсэн *entity*-ийг симуляц хийнэ гэдгийг заасан байна. Дараа нь *structure, signals, wave* цонхнуудыг нээх командууд өгөгджээ. Тэгээд оролтын сигналуудын стимули ямар байхыг заажээ. Эцэст нь 240нс хүртэл симуляцийг ажиллуулах командыг өгсөн байна. Ингэж макро файл үүсгэж ажиллуулсанаар энэ олон командуудыг бичиж, эсвэл меню маусны тусламжтай олон дахин өгч ажиллуулах гар ажиллагааг хөнгөвчилж байгаа сайн талтай юм.

2. Жишээ кодуудын макро файл үүсгэн ажиллагааг шалгаж харуулсан waveform-уудыг гарган авч макро файлтайгаа хамт тайландаа хавсарга.

3. Дээрх жишээтэй адилаар *JK* флип-флоп –ийн дараахь хувилбаруудын кодыг бич.

   - Асинхрон reset-тэй *JK FF (positive edge trigerred)*
   - Синхрон reset-тэй *JK FF (positive edge trigerred)*
   - Асинхрон reset болон preset-тэй *JK FF (positive edge trigerred)*
   - Асинхрон reset болон preset-тэй *JK FF (negative edge trigerred)*

4. Макро файл үүсгэн дээрх кодынхоо ажиллагааг шалгаж үр дүнг тайландаа хавсарга.