Лабораторийн ажил №2

Техник дүрслэлийн хэл
VHDL

Зорилго: *Тоон хэлхээг VHDL хэл дээр янз бүрийн аргаар дүрслэн кодолж болохыг судална. Тодруулбал behavorial, structural, dataflow төрлийн загваруудыг үзнэ.*

"Анхны тоо" бүртгэгчийн жишээ код

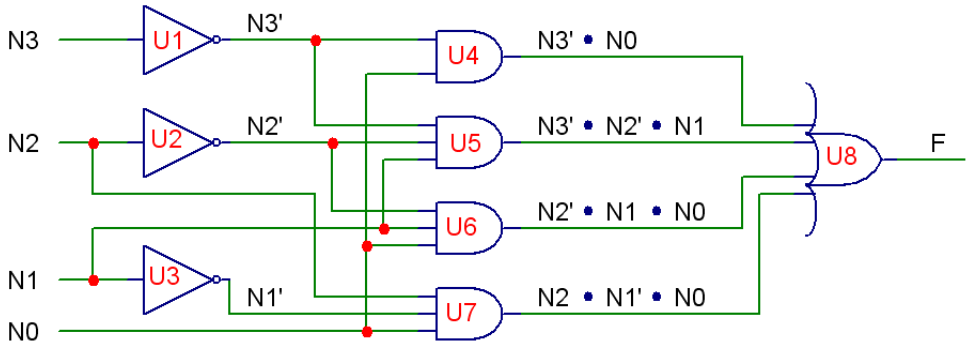
VHDL хэлээр тоон хэлхээний дизайн, түүний дүрслэлийг хийхэд үндсэн 3 хэлбэрээр гүйцэтгэж болдог. Үүнд:

- *behavioral*
- *structural*
- *dataflow*[1]

гэсэн дүрслэх аргууд байна. Энэхүү ажлаар 15 хүртэлх тооноос "анхны тоо"-г бүртгэдэг тоон схемийн дизайныг жишээ болгон дээрх гурван аргаар хэрхэн хийж гүйцэтгэж болохыг авч үзье.

Structural *хэлбэр*

Анхны тоо бүртгэдэг хэлхээний гэйт түвшиний схемийг үнэний хүснэгтээс нь хэрхэн гаргаж авдагийг бид урьд үзсэн "Тоон Систем" хичээлээс мэдэх билээ. Гэйт түвшин дэх логик схемийг дараахь зурагт үзүүлэв.



Зургаас харахад хэлхээ нь инвертерүүд, 2 болон 3 оролттой AND, мөн 4 оролттой OR гэйтүүд, тэдгээрийн хоорондын холбоосоос тогтсон байна. Энэхүү хэлхээг ашиглан бид өөрсдийн жишээг *structural* хэлбэртэйгээр VHDL дээр хийж гүйцэтгэх болно. Ер нь *structural VHDL* – ийн хэлбэр нь хэлхээний схематик буюу netlist-тэй ижил юм. Мэдээж *entity* хэсэг нь хэлхээний оролт, гаралтыг зарлаж өгнө. *Architecture* -даа эхлээд компонент болгон хэрэглэх гэж байгаа гэйтийнхээ төрлийг тодорхойлж өгнө. Нэгэнт бид архитектураа гэйтүүдээр хийх гэж байгаагаас хойш энд хэрэглэгдэх инвертерүүд, 2 болон 3 оролттой AND, мөн 4 оролттой OR гэйтүүдийг компонент гэж тодорхойлно. Архитектур доторх компонентүүд нь урьд нь дизайны нэг хэсэг гэж тодорхойлогдсон эсвэл *library*-ийн хэсэг байх ёстой. Манай тохиолдолд эдгээр гэйтүүд нь *lab2_gates.vhd* нэртэй package –д бичигдсэн байгаа бөгөөд үүнийг урьдчилан хөрвүүлсэн байх хэрэгтэй. Ингээд анхны тоо бүртгэгчийн *structural* VHDL загварыг дараахь байдлаар бичиж болно.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity prime1 is
   port ( N:  in STD_LOGIC_VECTOR(3 DOWNTO 0);
```

```
          F: out STD_LOGIC );
end prime1;

architecture prime_arch_struct of prime1 is
signal N3_L, N2_L, N1_L: STD_LOGIC;
signal N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0: STD_LOGIC;
component INV port(In1: in STD_LOGIC; Out1: out STD_LOGIC); end component;
component AND2 port(In1,In2: in STD_LOGIC; Out1: out STD_LOGIC); end component;
component  AND3  port(In1,In2,In3:  in  STD_LOGIC;  Out1:  out  STD_LOGIC);  end
component;
component OR4 port(In1,In2,In3,In4: in STD_LOGIC; Out1: out STD_LOGIC);
 end component;
begin
  U1: INV port map (N(3),N3_L);
  U2: INV port map (N(2),N2_L);
  U3: INV port map (N(1),N1_L);
  U4: AND2 port map (N3_L, N(0), N3L_N0);
  U5: AND3 port map (N3_L, N2_L, N(1), N3L_N2L_N1);
  U6: AND3 port map (N2_L, N(1), N(0), N2L_N1_N0);
  U7: AND3 port map (N(2), N1_L, N(0), N2_N1L_N0);
  U8: OR4 port map (N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0, F);
end prime_arch_struct;
```

Ингэж компонентүүд хэрэглэсэн VHDL архитектурыг *structural* дүрслэл буюу *structural* дизайн гэж нэрлэдэг.

***Dataflow** хэлбэр*

Энд хэлхээг нийлмэл бүтэц (structural) гэхээсээ илүү функц утгаар нь дүрсэлдэг бөгөөд түүнийг нэгэн зэрэг хийгдэх (concurrent) үйлдлүүд буюу тэдгээрийн адилтгалуудаар хийсэн байна. Хийх үйлдлүүд нь нэгэн хугацаанд буюу параллель гүйцэтгэгдэх ба өөрөөр хэлбэл тухайн операторын (statement) баруун гар тал дахь утгуудын нэг нь л өөрчлөгдөхөд бусад хэсэг нь мөн параллель гүйцэтгэгдэнэ гэсэн үг юм. Жишээ нь: Булийн тэгшитгэлийн баруун гар талын нэг утга өөрчлөгдөхөд л зүүн гар талынх нь харгалзан тооцогдон бодогдоно. VHDL-ийн хэд хэдэн нэгэн зэрэг хийгдэх (concurrent) тодорхойлох операторууд нь хэлхээн дэх өгөгдлийн урсгал болон түүн дээрх үйлдлүүдээр ямар нэг хэлхээг дүрслэх боломжтой болгодог. Ийм хэлбэрийг *dataflow* дүрслэл буюу *dataflow* дизайн гэж нэрлэдэг.
15 хүртэлх тооны анхны тоог бүртгэгчийн *dataflow* VHDL кодыг дараахь байдлаар үзүүлэв.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity prime2 is
   port ( N:  in STD_LOGIC_VECTOR(3 DOWNTO 0);
          F: out STD_LOGIC );
end prime2;

architecture prime1_arch_datafl of prime2 is
begin
  with N select
    F <= '1' when "0001",
         '1' when "0010",
         '1' when "0011" | "0101" | "0111",
         '1' when "1011" | "1101",
         '0' when others;
end prime1_arch_datafl;
```

Энд *select-when* гэсэн сонгогдсон сигналаар гүйцэтгэгдэх операторыг хэрэглэсэн бөгөөд өөрөөр бас *when-else* нөхцөлт оператор хэрэглэн дараахь байдлаар бичиж болно.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity prime3 is
   port ( N:  in STD_LOGIC_VECTOR(3 DOWNTO 0);
          F: out STD_LOGIC );
end prime3;

architecture prime2_arch_datafl of prime3 is
signal N3L_N0, N3L_N2L_N1, N2L_N1_N0, N2_N1L_N0: STD_LOGIC;
begin
  N3L_N0     <= '1' WHEN N(3)='0' and N(0)='1' ELSE '0';
  N3L_N2L_N1 <= '1' WHEN N(3)='0' and N(2)='0' AND N(1)='1' ELSE '0';
  N2L_N1_N0  <= '1' WHEN N(2)='0' and N(1)='1' AND N(0)='1' ELSE '0';
  N2_N1L_N0  <= '1' WHEN N(2)='1' and N(1)='0' AND N(0)='1' ELSE '0';
  F <= N3L_N0 or N3L_N2L_N1 or N2L_N1_N0 or N2_N1L_N0;
end prime2_arch_datafl;
```

***Behavioral** хэлбэр*

Логик түвшинээс нь дээш түвшинд хэлхээг дүрслэхийг ***behavioral*** түвшин буюу зарим тохиолдолд *register transfer level* (RTL) гэдэг. Заримдаа хийх гэж байгаа логик схемийнхээ ажиллагааг (юу хийхийг буюу behavior-ийг) concurrent команд ашиглан шууд бичих боломжтой байдаг. Энэхүү *behavioral* дизайн буюу дүрслэлийг хийх чадвар нь техник дүрслэлийн хэлний, тэр тусмаа VHDL –ийн нэг гол ашигтай тал юм. VHDL-ийн гол *behavioral* элемент нь "*process*" юм. *Process* нь "цуваа" (sequintial) тодорхойлогчуудын цуглуулга бөгөөд бусад concurrent тодорхойлогчууд болон бусад process-уудтай параллелиар гүйцэтгэгдэнэ. Өөрөөр хэлбэл *process* дотроо "цуваа" агуулгатай байна. *VHDL process* нь concurrent тодорхойлогчууд хэрэглэгдэж байгаа аль ч газар хэрэглэгдэнэ. *Process* нь *signal* зарлахгүй зөвхөн *variable* зарлана. Variable нь *process* дотор төлөвөө хадгалах ба process-ийн гадна харагдахгүй.
"Анхны тоо" бүртгэгчийн кодыг ***behavioral*** хэлбэрээр бичвэл:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity prime4 is
   port ( N:  in STD_LOGIC_VECTOR(3 DOWNTO 0);
          F: out STD_LOGIC );
end prime4;

architecture prime_arch_beh of prime4 is

function CONV_INTEGER (X: STD_LOGIC_VECTOR) return INTEGER is
  variable RESULT: INTEGER;
  begin
   RESULT:=0;
   for i in X'range loop
     RESULT := RESULT * 2;
     case X(i) is
       when '0' | 'L' => null;
       when '1' | 'H' => RESULT := RESULT + 1;
       when others  => null;
     end case;[3]
   end loop;
  return RESULT;
end CONV_INTEGER;

begin
  process(N)
```

```
    variable NI: integer;
    begin
     NI := CONV_INTEGER(N);
     if NI=1 OR NI=2 THEN F <= '1';
     elsif NI=3 OR NI=5 OR NI=7 OR NI=11 OR NI=13 then F <= '1';
     else F <= '0';
     end if;
  end process;
end prime_arch_beh;
```

Энд зөвхөн ганц concurrent команд байгаа ба тэр нь нэг *process* юм. Процессын *sensitivity list*-д N –ийг оруулсан бөгөөд энэ нь хийх гэж байгаа комбинац логикийн маань оролт юм. NI variable нь N оролтын *integer* рүү хөрвүүлэгдсэн утгыг барьж байхад хэрэглэгдэнэ. Ингэснээр *if* команд доторх харьцуулалт *integer* утгуудыг хэрэглэн хийгдэх боломжтой болсон. Энд *CONV_INTEGER* нэртэй функц хэрэглэгдэж байгаа бөгөөд *STD_LOGIC_VECTOR* төрлийг *INTEGER* болгон хөрвүүлэх үүрэг гүйцэтгэж байна.

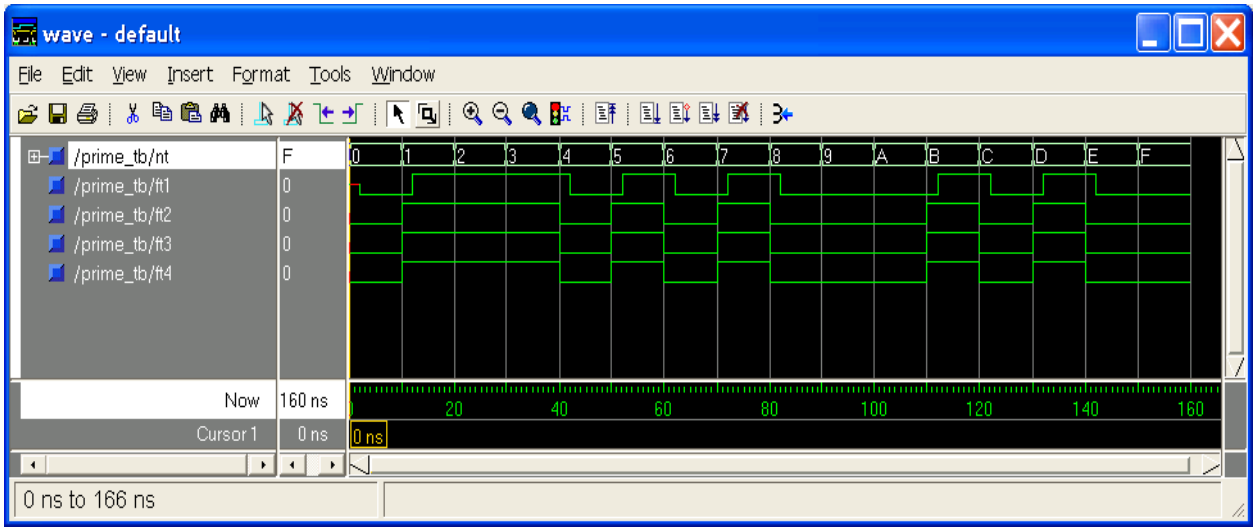
Даалгавар

1. Дээрх “анхны тоо” бүртгэгч кодуудын бүх хувилбарыг ModelSim симулятор дээр шалгаж үзээд үр дүнг харуул. Мэдээж шалгахын тулд testbench програмын кодыг бичих хэрэгтэй. Үүнийг дараахь байдлаар хийж болохыг харуулав.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity prime_tb is
end prime_tb;

architecture prime_tb_arch of prime_tb is

component prime1 port ( N:  in STD_LOGIC_VECTOR(3 DOWNTO 0);
           F: out STD_LOGIC ); end component;
component prime2 port ( N:  in STD_LOGIC_VECTOR(3 DOWNTO 0);
           F: out STD_LOGIC ); end component;
component prime3 port ( N:  in STD_LOGIC_VECTOR(3 DOWNTO 0);
           F: out STD_LOGIC ); end component;
component prime4 port ( N:  in STD_LOGIC_VECTOR(3 DOWNTO 0);
           F: out STD_LOGIC ); end component;

signal NT :STD_LOGIC_VECTOR(3 DOWNTO 0);
signal FT1, FT2, FT3, FT4 :STD_LOGIC;
begin
  UU1: prime1 port map (NT, FT1);
  UU2: prime2 port map (NT, FT2);
  UU3: prime3 port map (NT, FT3);
  UU4: prime4 port map (NT, FT4);
  process
  begin
    NT <= "0000";
    wait for 10 ns;
    NT <= "0001";[4]
    wait for 10 ns;[5]
    NT <= "0010";
    wait for 10 ns;
    NT <= "0011";
    wait for 10 ns;
    NT <= "0100";
    wait for 10 ns;
    NT <= "0101";
    wait for 10 ns;
```

```
    NT <= "0110";
    wait for 10 ns;
    NT <= "0111";
    wait for 10 ns;
    NT <= "1000";
    wait for 10 ns;
    NT <= "1001";
    wait for 10 ns;
    NT <= "1010";
    wait for 10 ns;
    NT <= "1011";
    wait for 10 ns;
    NT <= "1100";
    wait for 10 ns;
    NT <= "1101";
    wait for 10 ns;
    NT <= "1110";
    wait for 10 ns;
    NT <= "1111";
    wait for 10 ns;
    wait;
  end process;
end prime_tb_arch;
```

Энэхүү testbench код нь *behavioral* болон *structural* хэлбэрийн холимог байгааг харж болно.
Testbench-ийн симуляцийг гүйцэтгэснээр гарч болох waveform-ийг доорх зурагт үзүүлэв.



2. Гарсан үр дүн болон түүнээс хийх дүгнэлтээ ажлын тайланд хавсарга.
3. Дээрх жишээтэй адилаар 15 хүртэлх тооны гуравт хуваагддаг тоог бүртгэдэг схемийн VHDL кодыг ***structural, dataflow, behavioral*** хэлбэрүүдээр бичиж, түүнийгээ ModelSim симулятор ашиглан ажиллагааг нь *testbench* кодоор шалгаж үр дүнг харуул.