VHDL кодын синтез

Нарантуяа Цэндсүрэн
Монгол Улсын Их Сургууль
Мэдээлэл технологи, электроникийн сургууль
21B1NUM0298@stud.num.edu.mn

I.АЖЛЫН ЗОРИЛГО

Xilinx фирмийн WebPack ISE програмын синтез модулийг ашиглан VHDL дээр бичсэн кодыг хэрхэн синтезлэж, абстраст дүрслэлээ хэрхэн бодит схем болгох боломжтойг судална. Үүнээс гадна синтезлэгдсэн схем маань бидний сонгосон CPLD/FPGA төхөөрөмжид хэрхэн багтаж, зохион байгуулагдаж болохыг үзнэ.

II. WebPack ISE програм хангамж

Xilinx фирмийн WebPack ISE програм нь хэрэглэгчийн сонгосон програмчлагддаг логик төхөөрөмж (CPLD/FPGA)-д нь дизайныг орчинг тааруулж, илүү дээр дизайны дамжлага ажиллагааны боломжийг олгодог.
Ерөнхийдөө FPGA болон CPLD-н хувьд дизайны дамжлага ажиллагаа нь адил байна. Дизайнер өөрөө дизайнаа схематик хэлбэрээр үү эсвэл VHDL, Verilog, юм уу ABEL хэл дээрх хэлбэрээр оруулах уу гэдгээ сонгох болно.
Симуляцийн хувьд WebPack ISE програм нь ModelSim симулятортой хамтарч ажилладаг. Энэхүү симуляторийг бид урьдах ажлууд дээрээ өргөн ашиглаж байгаа тул сайн мэдэх билээ. WebPack ISE програм дээр тест хийх өгөгдлүүдийг (stimuli) харагдахуйцаар үүсгэхдээ хэрэглэгчийн график интерфэйс (Graphical User Interface) хэрэглэн хялбар хийх боломжтой. Ингэснээр testbench үүсгэн түүнийгээ тестлэх дизайнтай нь хамт ModelSim рүү хөрвүүлнэ.
Кодлох дизайн дуусаад, дизайнер симуляцийн үр дүнд хангалуун байвал, дизайнаа шаардагдах төхөөрөмж (CPLD/FPGA) рүүгээ чиглүүлнэ.
FPGA дээр хийх гүйцэтгэл процесс нь 4 шаттай байна.

1. Translate - энэ үед дизайныг хөрвүүлж дүрмийн шалгалт (design rule check) явуулна.

2. Map - хэрэглэгдэх гэж байгаа чип дэх нөөцийг тооцоолж, тодотгож авна.

3. Place and Route - зохимжтой байрлал дахь CLB (configure logic blocks)-ууд буюу тохируулагддаг логик блокуудыг авч, тэдгээрийг холбохдоо холбоос шугамуудын нөөцийг хэрэглэнэ.

4. Configure - програмчлах битийн урсгалыг үүсгэнэ.

II. Даалгавар

1. Хагас битийн нэмэгч

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

entity hagas_nemegc is
    Port ( x : in  STD_LOGIC;
           y : in  STD_LOGIC;
           s : out  STD_LOGIC;
           c : out  STD_LOGIC);
end hagas_nemegc;

architecture arch_ha of hagas_nemegc is
 begin
  s <= x xor y;
  c <= x and y;
end arch_ha;
```

Код 1: Хагас битийн нэмэгч

```
NET "x" LOC = "L13";
NET "y" LOC = "L14";
NET "s" LOC = "E12";
NET "c" LOC = "F12";
```

Код 2: UCF

1. Бүтэн битийн нэмэгч

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

entity full_adder is
    Port ( x : in  STD_LOGIC;
           y : in  STD_LOGIC;
           z : in  STD_LOGIC;
           s : out  STD_LOGIC;
           c : out  STD_LOGIC);
end full_adder;

architecture arch_fulladder of full_adder is
 component hagas_nemegc
  port (x, y: in std_logic;
      s, c: out std_logic);
 end component;
 signal hs, hc, tc: std_logic;
 begin
  HA1: hagas_nemegc port map (x, y, hs, hc);
  HA2: hagas_nemegc port map (hs, z, s, tc);
  c <= tc or hc;
end arch_fulladder;
```

Код 3: Нэг битийн бүтэн нэмэгч

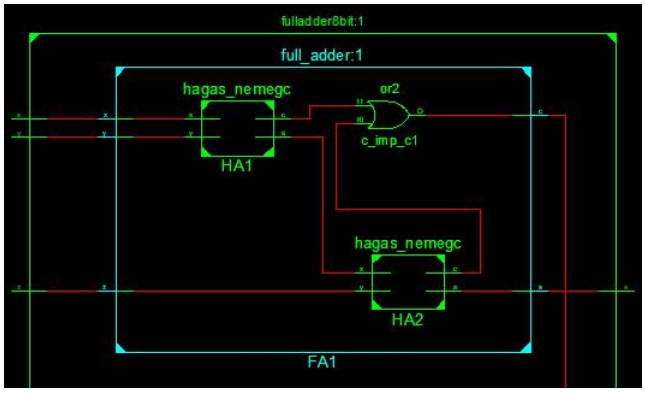
```
NET "x" LOC = "L13";
NET "y" LOC = "L14";
NET "s" LOC = "E12";
NET "c" LOC = "F12";
NET "z" LOC = "H18";
```

Код 4: UCP

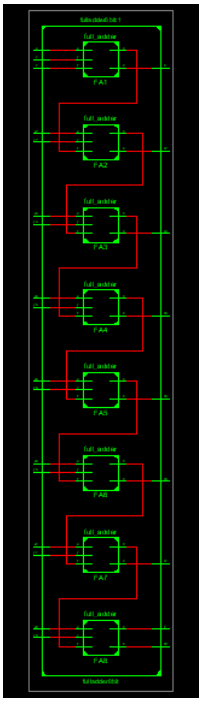
8 битийн бүтэн нэмэгч

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

entity fulladder8bit is
    Port ( x, x2, x3, x4, x5, x6, x7, x8 : in
    STD_LOGIC;
           y, y2, y3, y4, y5, y6, y7, y8 : in
    STD_LOGIC;
           z : in  STD_LOGIC;
           s, s2, s3, s4, s5, s6, s7, s8 : out
    STD_LOGIC;
           c : out  STD_LOGIC);
end fulladder8bit;

architecture arch_8bitfa of fulladder8bit is
component full_adder
  port (x, y, z: in std_logic;
      s, c: out std_logic);
end component;
signal fc, fc2, fc3, fc4, fc5, fc6, fc7, fc8:
    std_logic;
begin
  FA1: full_adder port map (x, y, z, s, fc);
  FA2: full_adder port map (x2, y2, fc, s2, fc2);
  FA3: full_adder port map (x3, y3, fc2, s3, fc3);
  FA4: full_adder port map (x4, y4, fc3, s4, fc4);
  FA5: full_adder port map (x5, y5, fc4, s5, fc5);
  FA6: full_adder port map (x6, y6, fc5, s6, fc6);
  FA7: full_adder port map (x7, y7, fc6, s7, fc7);
  FA8: full_adder port map (x8, y8, fc7, s8, c);
end arch_8bitfa;
```

Код 5: 8 битийн бүтэн нэмэгч



Зураг 1: RTL



Зураг 2: RTL

16 битийн бүтэн нэмэгч

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

entity fulladder16bit is
    Port ( x     : in  STD_LOGIC_VECTOR(15 downto 0)
    ;
           y     : in  STD_LOGIC_VECTOR(15 downto 0)
    ;
           z     : in  STD_LOGIC;
           s     : out STD_LOGIC_VECTOR(15 downto 0)
    ;
           c     : out STD_LOGIC);
end fulladder16bit;

architecture arch_16bitfa of fulladder16bit is
    component full_adder
        port ( x   : in  std_logic;
               y   : in  std_logic;
               z   : in  std_logic;
               s   : out std_logic;
               c   : out std_logic);
    end component;

    signal carry : std_logic_vector(15 downto 0);
begin
    gen_adders: for i in 0 to 15 generate
        adder_inst: full_adder
            port map (
                x => x(i),
                y => y(i),
                z => (carry(i-1) when i > 0 else z)
    ,
```

```
                s => s(i),
                c => carry(i)
            );
    end generate;
    c <= carry(15);
end arch_16bitfa;
```

Код 6: 16 битийн бүтэн нэмэгч

III. Дүгнэлт

ХАгач нэмэгчийг хийн, энэ компонентээ ашиглан бүтэн нэмэгчийг хийсэн. 4 бит, 8 бит, 16 битийн нэмэгчийг бүтэн нэмэг компонентээ ашиглан хийсэн.