Lab 1: Mạch logic tổ hợp và kịch bản kiểm tra (testbench)

Yêu cầu báo cáo:

Trong báo cáo nên có các phần sau:

- Source code VHDL của các phần thực hành (những phần phải viết code)
- Kết quả chạy mô phỏng, giải thích kết quả (dưới dạng dễ hiểu và dễ thấy, hình quá nhỏ hoặc chất lượng quá kém sẽ không được tính điểm. Trình bày đẹp sẽ được cộng điểm trình bày)
- Trả lời các câu hỏi trong phần hướng dẫn (phần lớn các phần có liên quan đến lý thuyết đã có trong slide môn học và trong giáo trình).
- Liên kết đến các phần lý thuyết đã học và những kiến thức mà các bạn đã học. Điểm cộng sẽ được tính cho các nội dung mở rộng (có thể có hoặc không có trong môn học) mà các bạn đã tìm hiểu.

Phần 1

Cho bảng chân lý như sau:

Input			Output
X	Y	S	m=F(X,Y,S)
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

Nhiệm vụ:

1. Viết phương trình logic của hàm F (dùng minterm hoặc maxterm).
2. Thực hiện các phương pháp tối giản logic (nếu có thể).
3. Hoàn thành chương trình VHDL miêu tả hoạt động của mạch logic tổ hợp được cho trong bảng chân lý:
   - Mở Terminal bằng cách vào Applications >> System Tools >> Terminal
   - Tạo thư mục lab1

```
mkdir -p $HOME/icdesign/m2
cp -a /home/tools/synopsys/m2/lab1 $HOME/icdesign/m2
cd $HOME/icdesign/m2/lab1
```

   - Mở tập tin mux21.vhd trong thư mục lab1/src với nội dung như sau:

```
LIBRARY ieee;
 USE ieee.std_logic_1164.ALL;

 ENTITY mux21 IS

   PORT (
```

```
    x : IN  STD_LOGIC;
    y : IN  STD_LOGIC;
    s : IN  STD_LOGIC;
    m : OUT STD_LOGIC);

END ENTITY mux21;

ARCHITECTURE df OF mux21 IS

BEGIN  -- architecture df

  m <= '0';                              -- to be completed

END ARCHITECTURE df;
```

- Viết phương trình của m dựa theo bảng chân lý bằng ngôn ngữ VHDL.

4. Chạy VCS và biên dịch thiết kế trong mục 3 bằng cách sử dụng các lệnh sau trong Terminal:

```
cd $HOME/icdesign/m2/lab1/sim
vhdlan -kdb ../src/mux21.vhd
vcs -kdb -debug_access+all mux21
./simv -ucli
```

Option -kdb để biên dịch mã nguồn dùng với chương trình Verdi của synopsys
-debug_access+all để bật tính năng gỡ rối thiết kế
-ucli để sử dụng chế độ dòng lệnh của chương trình mô phỏng. Để sử dụng chế độ đồ họa dùng option -gui để thay thế cho -ucli

5. Xuất tập tin dạng sóng và trích xuất tín hiệu để chạy mô phỏng

```
dump -file mux21.fsdb -type fsdb
dump -add / -aggregates
```

6. Tạo tín hiệu đầu vào dùng lệnh force:

```
force <signal name> <value> <time>, <value> <time>, ...
```

Ví dụ:

```
force X 0 0ns, 1 20ns, 0 50ns
force Y 0 0ns, 1 10ns, 0 60ns
force S 0 0ns, 1 40ns
```

7. Chạy mô phỏng:

```
run <time>
```

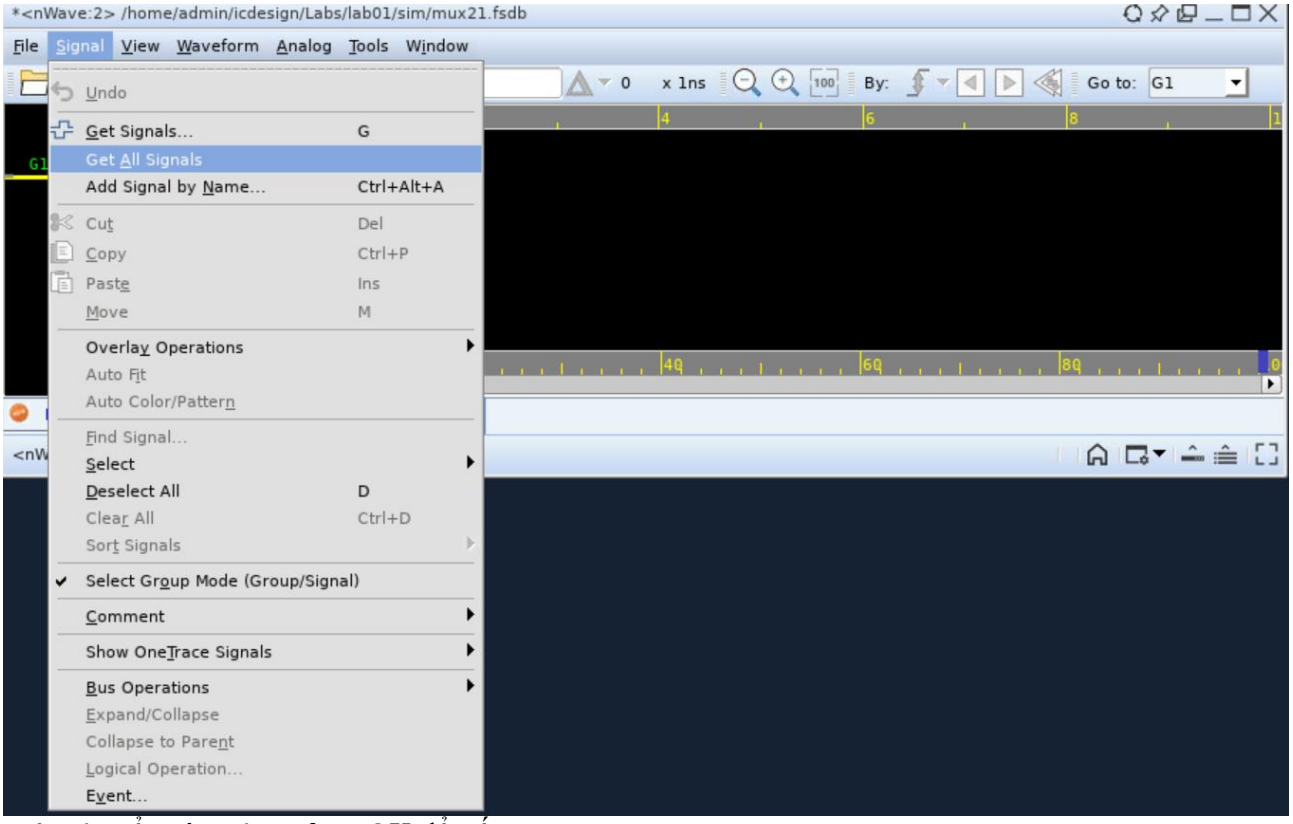
Ví dụ:

```
run 100ns
exit
```

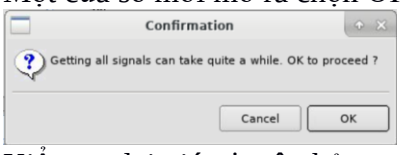
8. Hiển thị dạng sóng của tín hiệu bằng cách sử dụng phần mềm Verdi hoặc Design Vision (DVE)

```
verdi -ssf mux21.fsdb
```

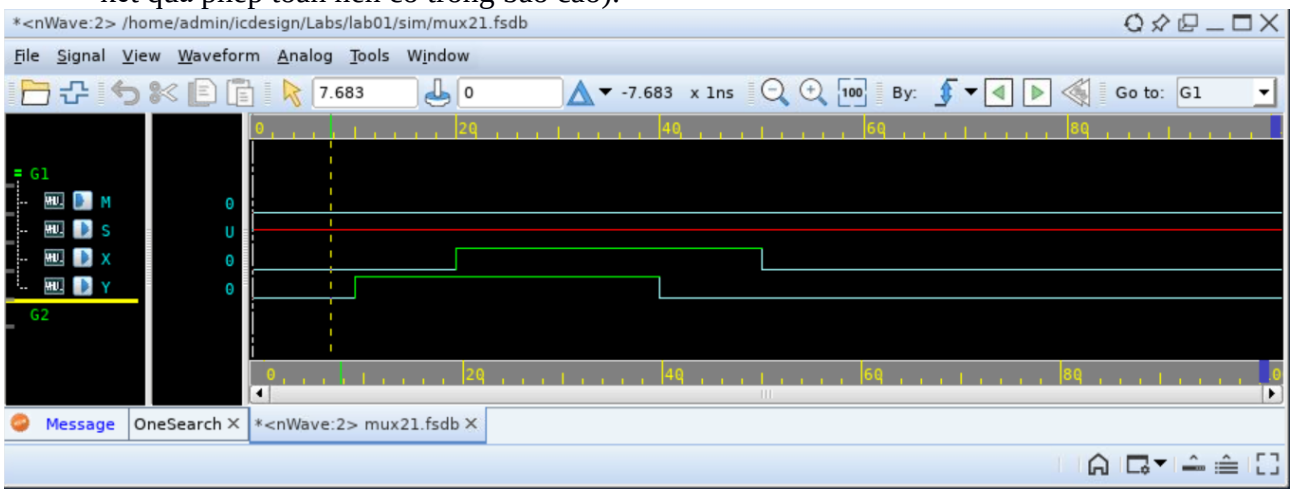
Trong cửa sổ dạng sóng, chọn `Signal >> Get all signals` để hiển thị dạng sóng



Một cửa sổ mới mở ra chọn OK để tiếp tục.



9. Kiểm tra lại giá trị mô phỏng có hợp với bảng chân lý bằng cách nhìn trong cửa sổ hiển thị dạng sóng. Kết quả mô phỏng (bao gồm cách tạo lệnh force và dạng sóng mô phỏng thể hiện kết quả phép toán nên có trong báo cáo).



Phần 2

1. Biến đổi phương trình logic trong phần 1 chỉ sử dụng phép toán logic (cổng logic) NAND.
2. Hoàn thành phần miêu tả kiến trúc dùng ngôn ngữ VHDL của mạch điện trong phần 1 chỉ sử dụng cổng logic NAND.

```
ARCHITECTURE df_nand OF mux21 IS
```

```
BEGIN  -- architecture df_nand

  m <= '0';                                -- to be completed

END ARCHITECTURE df_nand;
```

3. Biên dịch, chạy mô phỏng và kiểm tra tính đúng đắn của thiết kế sử dụng các câu lệnh đã học trong phần 1. Mã nguồn VHDL, kết quả mô phỏng và các lệnh thực hiện cần đưa vào trong báo cáo.

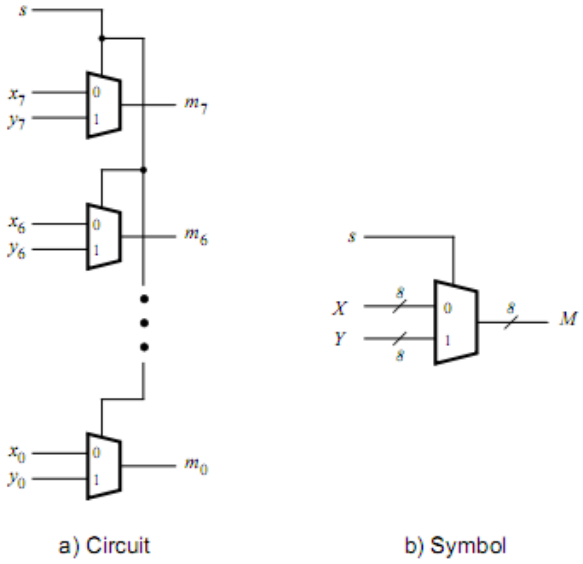
   Chú ý: chạy mô phỏng với kiến trúc df_nand bằng lệnh sau:

```
vhdlan -kdb mux21.vhd
vcs -kdb -debug_access+all work.mux21__df_nand
./simv -ucli -do scripts/mux21.tcl
```

Sau đó, dùng Verdi để mở tập tin mux21.fsdb để xem dạng sóng và so sánh với trường hợp trước.

Phần 3

Sử dụng thiết kế mux21 để thiết kế bộ hợp kênh với 2 đầu vào 8-bit mux21_8bit với sơ đồ như sau:



Hình 1: (a) Sơ đồ mạch điện để xây dựng bộ mux21_8bit từ 8 bộ mux21 với đầu vào 1 bit. (b) Sơ đồ ký hiệu của bộ mux21_8bit.

1. Viết chương trình VHDL để thực hiện mạch điện này sử dụng mẫu dưới đây:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux21_8bit IS

  PORT (
    x : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
    y : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
    s : IN  STD_LOGIC;
    m : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));

END ENTITY mux21_8bit;
```

```
ARCHITECTURE df OF mux21_8bit IS
  COMPONENT mux21 IS
    PORT (
      x : IN  STD_LOGIC;
      y : IN  STD_LOGIC;
      s : IN  STD_LOGIC;
      m : OUT STD_LOGIC);
  END COMPONENT mux21;
BEGIN  -- ARCHITECTURE df
  mux21_0 : mux21
    PORT MAP (
      x => '0',                                 -- to be completed
      y => '0',                                 -- to be completed
      s => '0',
      m => OPEN);

  mux21_1 : mux21
    PORT MAP (
      x => '0',
      y => '0',
      s => '0',
      m => OPEN);
-- continue for other mux21 instances
-- there should be 8 instances of mux21
END ARCHITECTURE df;
```

2. Chạy mô phỏng sử dụng các câu lệnh như trong phần 1. Kiểm tra tính đúng đắn của thiết kế. Giải thích kết quả mô phỏng (mã nguồn VHDL và dạng sóng trong quá trình chạy mô phỏng phải được đưa vào trong báo cáo).
3. Hoàn thành kịch bản kiểm tra (testbench) của bộ mux21_8bit sử dụng mẫu dưới đây:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux21_8bit_tb IS

END ENTITY mux21_8bit_tb;

ARCHITECTURE test OF mux21_8bit_tb IS

  -- component ports
  SIGNAL x_net : STD_LOGIC_VECTOR(7 DOWNTO 0);
  SIGNAL y_net : STD_LOGIC_VECTOR(7 DOWNTO 0);
  SIGNAL s_net : STD_LOGIC;
  SIGNAL m_net : STD_LOGIC_VECTOR(7 DOWNTO 0);

  CONSTANT delay : TIME := 10ns;

BEGIN  -- ARCHITECTURE test

  -- component instantiation
  DUT : ENTITY work.mux21_8bit
```

```
    PORT MAP (
      x => x_net,
      y => y_net,
      s => s_net,
      m => m_net);

  -- waveform generation
  WaveGen_Proc : PROCESS
  BEGIN
    -- insert signal assignments here
    x_net <= (OTHERS => '0');
    y_net <= (OTHERS => '0');
    s_net <= '0';
    WAIT FOR delay;
  END PROCESS WaveGen_Proc;
END ARCHITECTURE test;
```

Thêm phần tự động kiểm tra kết quả đúng hoặc sai trong kịch bản kiểm tra dùng lệnh ASSERT.

4. Chạy mô phỏng thiết kế. Giải thích kết quả chạy mô phỏng. Kịch bản kiểm tra của bạn có bao gồm tất cả các khả năng của tín hiệu đầu vào?
   (Mã nguồn của kịch bản kiểm tra và dạng sóng của phần mô phỏng phải được đưa vào trong báo cáo).
5. Nêu sự khác nhau trong phong cách thiết kế và miêu tả hành vi của mạch điện sử dụng ngôn ngữ VHDL trong phần 1, phần 2 và phần 3. Theo bạn, những phong cách thiết kế này có ưu điểm và nhược điểm gì và khi nào thì nên sử dụng chúng.
6. Bonus: Viết kịch bản tự động kiểm tra tính đúng đắn của thiết kế.

Phần 4

1. Bộ mux21 của hai số n-bit có thể được thiết kế bằng cách sử dụng các bộ mux21 dùng 1-bit đã được thiết kế và kiểm tra. Để thiết kế bộ mux21 của 2 số n-bit với số n thay đổi tùy thuộc vào nhu cầu của người sử dụng, chúng ta phải sử dụng khai báo *GENERIC* của ENTITY để tham số hóa tham số n và sử dụng *FOR-GENERATE* để tạo ra cấu trúc phần cứng như trong Hình 1 (a). Sử dụng tập tin mux21_nbit.vhd để hoàn thành thiết kế bộ mux21 của 2 số n-bit từ các bộ mux21 1-bit sử dụng cấu trúc *FOR-GENERATE*.
   (Mã nguồn phải được đưa vào trong báo cáo).

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux21_nbit IS

  GENERIC (
    DATA_WIDTH : integer := 16);

  PORT (
    X : IN  STD_LOGIC_VECTOR(DATA_WIDTH-1 DOWNTO 0);
    y : IN  STD_LOGIC_VECTOR(DATA_WIDTH-1 DOWNTO 0);
    s : IN  STD_LOGIC;
    m : OUT STD_LOGIC_VECTOR(DATA_WIDTH-1 DOWNTO 0));

END ENTITY mux21_nbit;
```

```
ARCHITECTURE df OF mux21_nbit IS
  COMPONENT mux21 IS
    PORT (
      x : IN  STD_LOGIC;
      y : IN  STD_LOGIC;
      s : IN  STD_LOGIC;
      m : OUT STD_LOGIC);
  END COMPONENT mux21;
BEGIN  -- ARCHITECTURE df

  mux21_gen: FOR i IN 0 TO DATA_WIDTH-1 GENERATE
    mux21_i: ENTITY work.mux21
      PORT MAP (
        x => '0',                          -- to be completed
        y => '0',                          -- to be completed
        s => '0',                          -- to be completed
        m => OPEN);                        -- to be completed
  END GENERATE mux21_gen;

END ARCHITECTURE df;
```

2. Viết kịch bản kiểm tra để chứng minh tính đúng đắn của thiết kế trong phần 4.1. (Mã nguồn của kịch bản kiểm tra và dạng sóng của phần mô phỏng phải được đưa vào trong báo cáo).

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

--------------------------------------------------------------------------------

ENTITY mux21_nbit_tb IS

END ENTITY mux21_nbit_tb;

--------------------------------------------------------------------------------

ARCHITECTURE testbench OF mux21_nbit_tb IS

  -- component generics
  CONSTANT DATA_WIDTH : integer := 16;
  CONSTANT delay : TIME := 10 NS;

  -- component ports
  SIGNAL x_net : STD_LOGIC_VECTOR(DATA_WIDTH-1 DOWNTO 0);
  SIGNAL y_net : STD_LOGIC_VECTOR(DATA_WIDTH-1 DOWNTO 0);
  SIGNAL s_net : STD_LOGIC;
  SIGNAL m_net : STD_LOGIC_VECTOR(DATA_WIDTH-1 DOWNTO 0);

BEGIN  -- ARCHITECTURE testbench

  -- component instantiation
  DUT: ENTITY work.mux21_nbit
    GENERIC MAP (
      DATA_WIDTH => DATA_WIDTH)
    PORT MAP (
      x => x_net,
      y => y_net,
      s => s_net,
      m => m_net);

  -- waveform generation
  WaveGen_Proc: process
```

```
  begin
    -- insert signal assignments here
    x_net <= (OTHERS => '0');
    y_net <= (OTHERS => '0');
    s_net <= '0';
    WAIT FOR delay;
  end process WaveGen_Proc;

END ARCHITECTURE testbench;

--------------------------------------------------------------------------------

CONFIGURATION mux21_nbit_tb_testbench_cfg OF mux21_nbit_tb IS
  FOR testbench
  END FOR;
END mux21_nbit_tb_testbench_cfg;

--------------------------------------------------------------------------------
```

3. Thay vì dùng cấu trúc FOR-GENERATE, ví dụ này có thể được thiết kế bằng cách dùng kiểu miêu tả dataflow. Hãy viết lại thiết kế trong phần 4.1 bằng kiểu miêu tả này (gợi ý: có thể dùng cấu trúc WHEN-ELSE) và chạy mô phỏng lại thiết kế dùng testbench được viết trong mục 4.2.

   (Chèn code và kết quả chạy mô phỏng và giải thích ý tưởng vào trong báo cáo)