

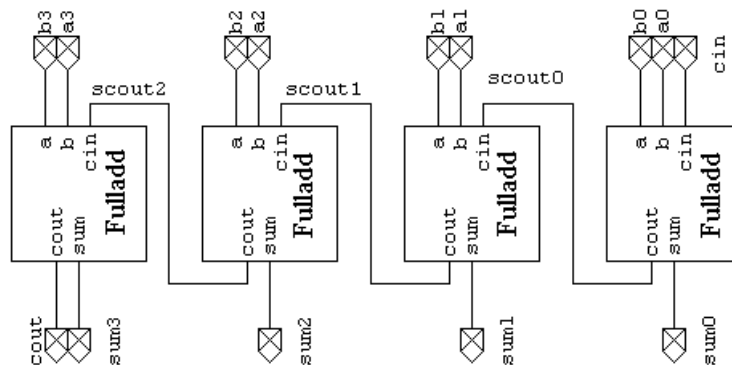
การทดลองที่ 2 4 Bit Ripple Adder

จุดประสงค์

1. เพื่อให้รู้จักการเขียนภาษาวีเอชดีแอลแบบ Structural
2. เพื่อให้มีความสามารถใช้ Testbench สำหรับทดสอบการทำงานของวงจรได้
3. สามารถใช้คำสั่งเกี่ยวกับไฟล์ได้

บทนำ

การทดลองที่ 1 ได้สร้างวงจรบวกเลขแบบเต็ม (Full adder) ขนาด 1 บิตขึ้น ในการทดลองที่ 2 นี้เราจะใช้วงจรบวกเลขนั้นเป็น อุปกรณ์เพื่อสร้างเป็นวงจรบวกเลขขนาด 4 บิตแบบรีปเปิล ตามรูปที่ 2.1 นี้



รูปที่ 2.1 วงจร 4 Bit Ripple adder

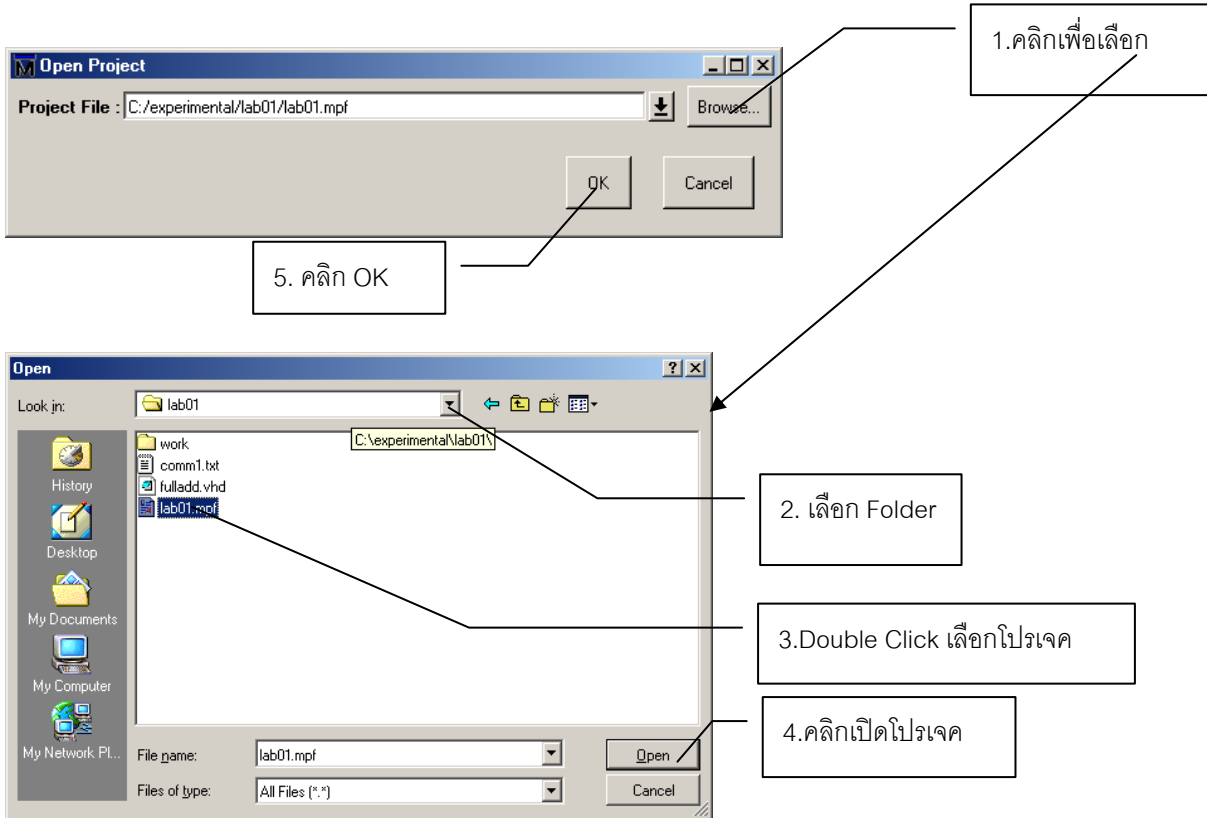
สำหรับรูปแบบการเขียนใช้ลักษณะ Structural ดังรูปที่ 2.2 นี้

```
entity adder4 is
    port (a : in bit_vector(3 downto 0);
          b : in bit_vector(3 downto 0);
          cin : in bit;
          sum : out bit_vector(3 downto 0);
          cout : out bit);
end adder4;
architecture adder4beh of adder4 is
    component fulladd
        port (a : in bit;
              b : in bit;
              cin : in bit;
              sum : out bit;
              cout : out bit);
    end component;
    signal scout : bit_vector( 2 downto 0);
begin
    c1 : fulladd port map (a(0), b(0), cin, sum(0), scout(0));
    c2 : fulladd port map (a(1), b(1), scout(0), sum(1), scout(1));
    c3 : fulladd port map (a(2), b(2), scout(1), sum(2), scout(2));
    c4 : fulladd port map (a(3), b(3), scout(2), sum(3), cout);
end adder4beh;
```

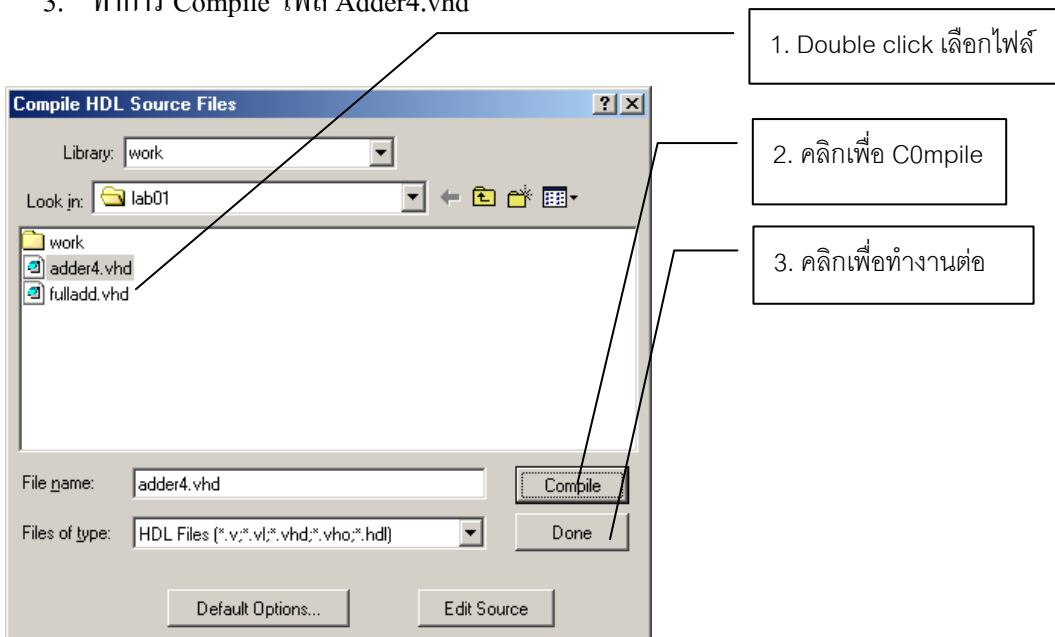
รูปที่ 2.2 VHDL Model วงจร 4 Bit Ripple adder

การทดลอง

1. ในกรณีที่ยังไม่ได้เปิดโปรแกรม ModelSim ให้ข้ามไปทำข้อ 2 แต่ถ้าเปิดแล้วให้เปิดโปรเจก Lab01 ขึ้นมาใหม่ โดยใช้คำสั่ง File > Open > Open Project



2. เขียนไฟล์ Adder4.vhd โดยใช้คำสั่ง File > New > New Source แล้วบันทึกไว้ใน Folder Lab01
3. ทำการ Compile ไฟล์ Adder4.vhd




หมายเหตุ ในขณะที่ไฟล์ Fulladd.vhd ต้องถูก Compile ไว้ก่อนแล้ว ถ้ายัง จะมี Error ที่บอกว่าไม่รู้จักอุปกรณ์ Fulladd

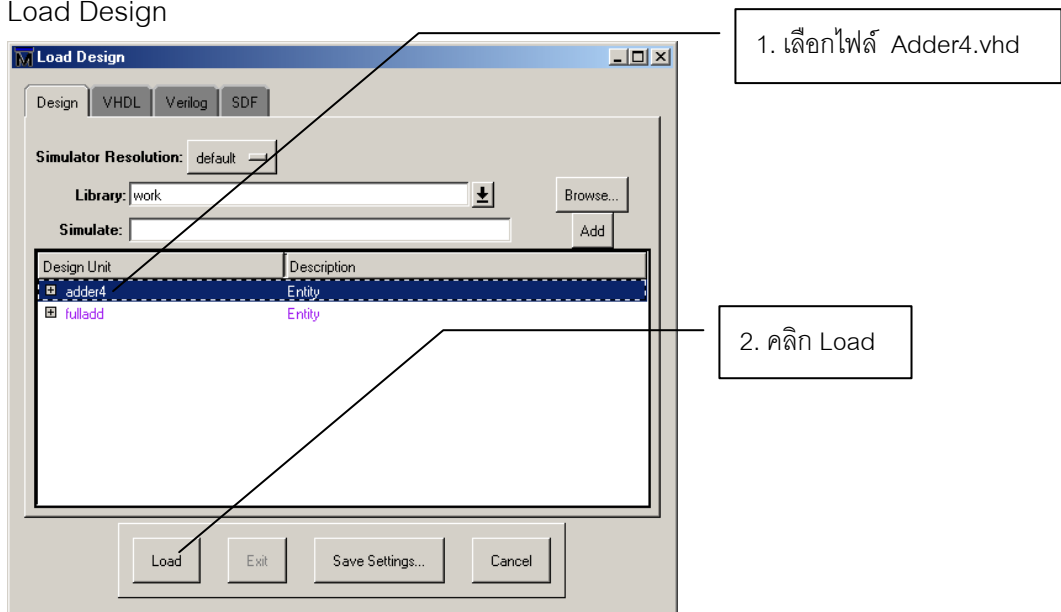
4. สร้างไฟล์มาโครให้มีคำสั่งต่อไปนี้แล้วบันทึกไว้ในชื่อ comm2.txt และไว้ใน folder เดียวกับ fulladd.vhd

```
view wave
view signals
add wave sim:/adder4/a
add wave sim:/adder4/b
add wave sim:/adder4/cin
add wave sim:/adder4/sum
add wave sim:/adder4/cout
force a 1100
force b 1011
force cin 0
run
force a 0011
force b 1000
force cin 1
run
```

รูปที่ 2.2 ไฟล์มาโคร

5. จำลองการทำงาน โดยใช้คำสั่ง Design > Load New Design หรือคลิก  จะปรากฏหน้าต่าง

Load Design



6. สั่งจำลองการทำงานด้วยคำสั่ง do comm2.txt ในหน้าต่าง Transcript

7. คุณผลที่ได้ในหน้าต่าง wave ถ้าคุณไม่ชดให้ใช้การขยายหน้าต่าง ตรวจสอบว่าข้อมูลถูกต้องหรือไม่ถ้าไม่ถูกต้องแสดงว่า การเขียนมีข้อผิดพลาดให้กลับไปแก้ไขแล้วทดลองใหม่

8. หยุดการจำลองการทำงานด้วยคำสั่ง Design > End Simulation

การใช้ Testbench แบบที่ 1

```
entity test1_adder4 is
end test1_adder4;
architecture tadder4beh of test1_adder4 is
component adder4
  port (a : in bit_vector(3 downto 0);
        b : in bit_vector(3 downto 0);
        cin : in bit;
        sum : out bit_vector(3 downto 0);
        cout : out bit);
end component;

signal sa,sb,ssum : bit_vector(3 downto 0);
signal scin, scout : bit;

begin
  adder : adder4 port map (sa, sb, scin, ssum, scout);
  test:
  process
    begin
      sa <= "0011";
      sb <= "1010";
      scin <= '0';
      wait for 100 ns;
      sa <= "1011";
      sb <= "0010";
      wait for 100 ns;
      sb <= "1111";
      wait;
    end process;
end tadder4beh;
```

รูปที่ 2.4 VHDL Model สำหรับ Testbench เพื่อใช้ตรวจสอบวงจร 4 Bit Ripple adder

การทดลอง

1. สร้างไฟล์ใหม่ตามรูปที่ 2.4 แล้วบันทึกไว้ในชื่อ test1_adder4.vhd
2. ทำการ Compile ไฟล์ Test1_add4.vhd
3. ทำการทดสอบโดย Load Design ของ test1_adder4
4. เปิดหน้าต่าง Signal
5. เปิดหน้าต่าง Wave
6. เลือกสัญญาณจากหน้าต่าง Signal ให้แสดงที่หน้าต่าง Wave
7. จำลองการทำงานโดยคำสั่ง run 200 ns
8. หยุดการจำลองการทำงาน โดยคลิกขวาพื้นที่ของ Sim tab แล้วเลือกคำสั่ง End Simulation แล้วตรวจสอบผลการทำงาน ให้ทดลองเปลี่ยนแปลงไฟล์ testbench แล้วเริ่มทดลองใหม่ตั้งแต่ข้อ 2
9. ถ้าผลการทำงานถูกต้องให้หยุดการจำลองการทำงานด้วย Design > End Simulation

การใช้ Testbench แบบที่ 2

การทดลองที่ผ่านมาใช้วิธีการป้อนข้อมูลอินพุตผ่านทางคีย์ หรือผ่านทาง testbench ซึ่งเป็นการไม่สะดวกสำหรับการเปลี่ยนแปลงค่าข้อมูล ในการทดลองนี้จะเก็บข้อมูลสำหรับอินพุตไว้ในไฟล์ชื่อว่า num.txt โดยมี ตัวอย่างไฟล์ดังนี้

```
0000 0000
0011 0101
0100 1010
1100 1110
0011 1100
```

รูปที่ 2.5 ตัวอย่างไฟล์อินพุต

และเมื่อทดสอบการทำงานแล้วผลลัพธ์ก็จะเขียนลงไฟล์ชื่อ num2.txt ดังนี้

```
0000
1000
1110
1010
1111
```

รูปที่ 2.6 ตัวอย่างไฟล์เอาต์พุต num2.txt

คำสั่งเกี่ยวกับไฟล์ในภาษา VHDL

VHDL สามารถติดต่อกับไฟล์ได้โดยใช้ Library std.textio โดยเขียนคำสั่ง use std.textio.ALL

ไฟล์ที่ใช้สำหรับการอ่านใช้เป็น FILE file_descriptor : text is IN "ชื่อไฟล์";

ไฟล์ที่ใช้สำหรับการเขียนใช้เป็น FILE file_descriptor : text is OUT "ชื่อไฟล์";

การทดลอง

1. เปิดโปรเจค Lab01 ถ้าเปิดอยู่แล้วก็ไม่ต้อง
2. สร้างไฟล์ใหม่ตามรูปที่ 2.7 แล้วบันทึกไว้ในชื่อ test2_adder4.vhd
3. ทำการ Compile ไฟล์ Test2_add4.vhd
4. ทำการทดสอบโดย Load Design ของ test2_adder4
5. เปิดหน้าต่าง Signal และหน้าต่าง Wave
6. ทำการจำลองการทำงาน โดยคำสั่ง run 190 ns
7. หยุดการจำลองการทำงาน
8. ตรวจสอบผลการทำงานที่ไฟล์ num2.txt ให้ทดลองเปลี่ยนแปลงค่าในไฟล์ num.txt แล้วเริ่มทดลองใหม่

```

use std.textio.all;
entity test2adder4 is
end test2adder4;
architecture tadder4beh of test2adder4 is
component adder4
  port (a : in bit_vector(3 downto 0);
        b : in bit_vector(3 downto 0);
        cin : in bit;
        sum : out bit_vector(3 downto 0);
        cout : out bit);
end component;

signal sa,sb,ssum : bit_vector(3 downto 0);
signal scin, scout : bit;

begin
  scin <= '0';
  adder : adder4 port map (sa, sb, scin, ssum, scout);
  test:
  process

      file vector_file : text is in "num.txt";
      file out_file : text is out "num2.txt";
      variable l : line;
      variable int : bit_vector(3 downto 0);
      begin
        while not endfile(vector_file) loop
          readline(vector_file, l);
          read(l, int);
          sa <= int;
          read(l,int);
          sb <= int;
          wait for 10 ns;
          write(l,ssum);
          writeline(out_file,l);
          wait for 30 ns;
        end loop;
      end process;
end tadder4beh;

```

รูปที่ 2.7 VHDL Model สำหรับการ Testbench เพื่อตรวจสอบวงจร 4 Bit Ripple adder โดยอ่านจากไฟล์