

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

Experiment Adv DSD-01: Introduction to CAD Tool & VHDL

วัตถุประสงค์: เพื่อเรียนรู้การใช้งานซอฟต์แวร์ช่วยในการออกแบบวงจรดิจิทัล และการออกแบบวงจรด้วยภาษา VHDL

บทนำ

การออกแบบวงจรดิจิทัลโดยใช้ FPGA สามารถทำได้หลายวิธี ได้แก่ วิธีวาดผังวงจร (Schematic) เขียนบรรยายการทำงานวงจรด้วยภาษา HDL (Hardware Description Language) หรือการเขียนแผนภูมิสถานะ (State diagram)

การออกแบบด้วยวิธีวาดผังวงจรจะเหมาะสำหรับผู้ที่เริ่มต้นเรียนรู้การออกแบบวงจรดิจิทัล ซึ่งสามารถออกแบบวงจรได้ทันทีโดยไม่ต้องศึกษาภาษาระดับสูงแต่อย่างใด แต่การออกแบบด้วยวิธีวาดผังวงจรมันไม่เหมาะกับการออกแบบวงจรดิจิทัลขนาดใหญ่ เนื่องจากใช้เวลาในการออกแบบค่อนข้างมากและการตรวจสอบข้อผิดพลาดเป็นไปได้ยากทำให้การออกแบบเกิดความล่าช้า การออกแบบวงจรด้วยภาษา HDL ซึ่งเป็นภาษาระดับสูง จึงเป็นการทุ่นเวลาให้รวดเร็วยิ่งขึ้นและง่ายต่อการตรวจสอบข้อผิดพลาด ภาษา HDL ที่นิยมใช้ในการออกแบบได้แก่ ภาษา Verilog และ VHDL เป็นต้น

CAD Tool

Computer-Aided Design (CAD) Tool คือเครื่องมือที่ช่วยในการออกแบบให้สะดวกสบายขึ้น สาขาที่มีการใช้ซอฟต์แวร์ประเภทมาก คือ การออกแบบทางวิศวกรรม เช่น การออกแบบสิ่งก่อสร้าง การออกแบบถนน การออกแบบเครื่องจักรกล รวมถึงการออกแบบงานทางไฟฟ้า

ISE WebPACK เป็นซอฟต์แวร์ประเภท CAD ที่ใช้ในงานออกแบบวงจรดิจิทัล ตั้งแต่การ Design, Testing จนถึงการ Debugging ซึ่งสามารถออกแบบได้ทั้งวิธีการวาดผังวงจร การออกแบบด้วยภาษา HDL และการออกแบบด้วย State diagram โดยในการทดสอบวงจรทำได้ในลักษณะของการ Simulate Timing Diagram ได้

VHDL

VHDL (VHSIC Hardware Description Language, VHSIC=Very High Speed Integrated Circuit) เป็นภาษาระดับสูงที่ใช้ในการออกแบบระบบและวงจรดิจิทัล โดยมีรูปแบบคล้ายกับภาษาปาสคาล เป็นภาษาที่สามารถปรับเปลี่ยนโครงสร้างให้เข้ากับระบบฮาร์ดแวร์ได้ สามารถออกแบบวงจรในระดับต่างๆได้หลายระดับ โดยที่การออกแบบระบบจะอยู่ในรูปแบบของฟังก์ชันเท่านั้น การเปลี่ยนแปลงแก้ไขวงจรหรือนำกลับมาใช้ใหม่จึงทำได้ง่าย มีซอฟต์แวร์ช่วยในการตรวจสอบความถูกต้องของวงจรที่

ออกแบบ (Verification) โดยการจำลองการทำงาน (Simulation) จึงไม่จำเป็นต้องทดสอบกับวงจรจริงและสามารถดูผลลัพธ์ของวงจรที่ออกแบบได้

VHDL พื้นฐาน

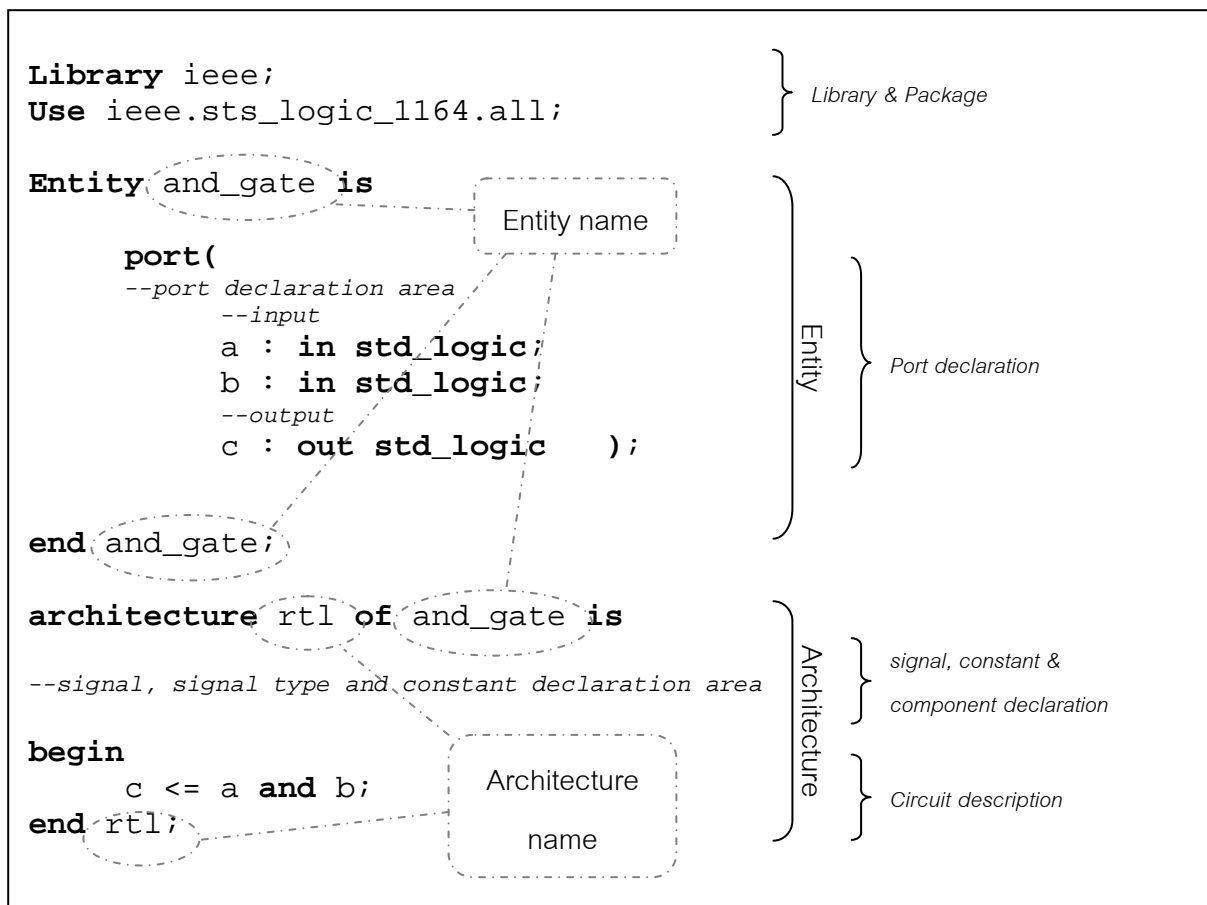
องค์ประกอบของภาษา VHDL มีหน่วยการออกแบบ (Design Unit) หลักๆ ได้แก่

- Entities
- Architecture
- Package (จะอธิบายในการทดลองครั้งถัดไป)
- Configuration (จะอธิบายในการทดลองครั้งถัดไป)

Entity คือส่วนที่เป็น Interface ที่บอกรายละเอียดอินพุต/เอาต์พุตพอร์ตของระบบดิจิทัลที่ออกแบบว่ามีการเชื่อมต่อกับวงจรมานอกอย่างไรบ้าง

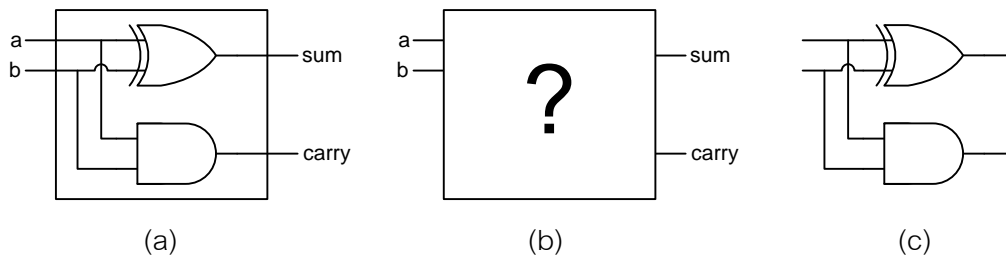
Architecture คือส่วนที่เป็น Body เป็นหน่วยออกแบบที่อธิบายฟังก์ชันการทำงานของระบบดิจิทัลที่ออกแบบ

โดยทั่วไปวงจรที่ออกแบบด้วย VHDL จะประกอบด้วยหน่วยการออกแบบพื้นฐานอย่างน้อย 2 หน่วย คือ Entity declaration และ Architecture body



รูปที่ 1.1 ตัวอย่างการเขียน VHDL พื้นฐาน

วงจรต่อไปนี้ ดังรูปที่ 1.2 เป็นตัวอย่างของการออกแบบด้วยภาษา VHDL ซึ่งเป็นวงจร Half-Adder หากมองภาพรวมของวงจรจะเป็นดังรูปที่ 1.2(a) ซึ่งมีส่วนประกอบของอินพุต/เอาต์พุตและลักษณะการทำงานของวงจร ส่วนของ Entity จะเป็นดังรูปที่ 1.2(b) ซึ่งจะบอกเฉพาะส่วนของอินพุต/เอาต์พุตที่ติดต่อกับภายนอก และรูปที่ 1.2(c) จะเป็นส่วน Architecture ที่อธิบายภายในของ Half-Adder ว่าทำงานอย่างไร



รูปที่ 1.2 วงจร Half-Adder

- (a) วงจรโดยรวมของ Half-Adder
- (b) ลักษณะ Entity ของ Half-Adder
- (c) ลักษณะ Architecture ของ Half-Adder

เมื่อนำวงจร Half-Adder มาเขียนด้วย VHDL จะได้ดังรูปที่ 1.3 จะเห็นว่า ภายใน entity จะมีการประกาศพอร์ต ว่าจุดใดบ้างเป็นอินพุต จุดใดบ้างเป็นเอาต์พุต ส่วน architecture จะอธิบายว่าภายใน Half-Adder ทำงานอย่างไรบ้าง

```

ENTITY half_adder IS
  PORT ( a      : IN BIT;
         b      : IN BIT;
         sum    : OUT BIT;
         carry  : OUT BIT);
END half_adder;

ARCHITECTURE rtl OF half_adder IS
BEGIN
  sum    <= a XOR b;
  carry  <= a AND b;
END rtl;
    
```

รูปที่ 1.3 วงจร Half-Adder ที่เขียนด้วยภาษา VHDL

ภายใน Architecture ทุกคำสั่งที่อยู่ภายใน ตั้งแต่ begin จนถึง end จะทำงานแบบ Concurrent นั่นคือทุกคำสั่งที่อยู่ภายในจะทำงานพร้อมกัน เมื่อเขียนโปรแกรมด้วยคำสั่ง ดังนี้

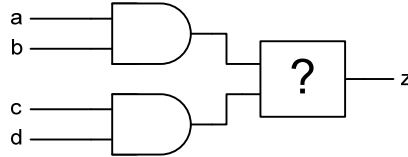
```

ARCHITECTURE rtl OF conccrrnt IS
    
```

```

BEGIN
    z <= a AND b;
    z <= c AND d;
END rtl;
    
```

(a)



(b)

รูปที่ 1.4 วงจรแบบ Concurrent

(a) การเขียนแบบ Concurrent

(b) วงจรที่ได้จากการเขียนแบบ Concurrent

วงจรถูกได้จะเป็นดังรูปที่ 1.3 ซึ่งเห็นได้ว่าวงจรถูกแบบนี้ไม่สามารถบอกได้ว่า เกตที่ติดกับเอาต์พุต Z คือเกตใด จึงทำให้การทำงานผิดพลาดขึ้นได้

ดังนั้น จึงต้องมีการกำหนดให้ทำงานแบบ Sequential ด้วยคำสั่ง Process ซึ่งจะทำให้ส่วนของโปรแกรมนั้นทำงานเป็นลำดับที่ละบรรทัด ดังรูปที่ 1.5

```

ARCHITECTURE rtl OF seq IS
BEGIN
    p1: PROCESS(a, b, c, d, z)
        BEGIN
            z <= a and b;
            z <= c and d;
        END p1;
END rtl;
    
```

(a)



(b)

รูปที่ 1.5 วงจรแบบ Sequential

(a) การเขียนแบบ Sequential


(b) วงจรที่ได้จากการเขียนแบบ Sequential

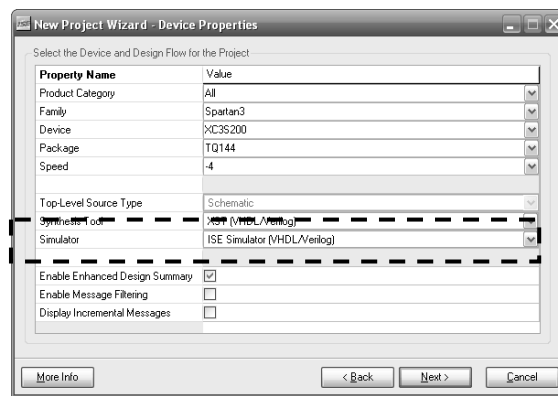
คำสั่งที่อยู่ใน Process จะทำงานเป็นลำดับที่ละบรรทัด ทำให้ทราบได้ว่าวงจรถูกที่เขียนขึ้นเป็นเช่นไร การทำงานแบบ Sequential จำเป็นสำหรับการทำงานจำพวก Condition เช่น If, When, Case เป็นต้น ซึ่งคำสั่งเหล่านี้ไม่สามารถทำงานได้ เมื่ออยู่ในสภาวะการทำงานแบบ Concurrent ได้

ใน architecture สามารถใช้ process ได้มากกว่าหนึ่งครั้ง

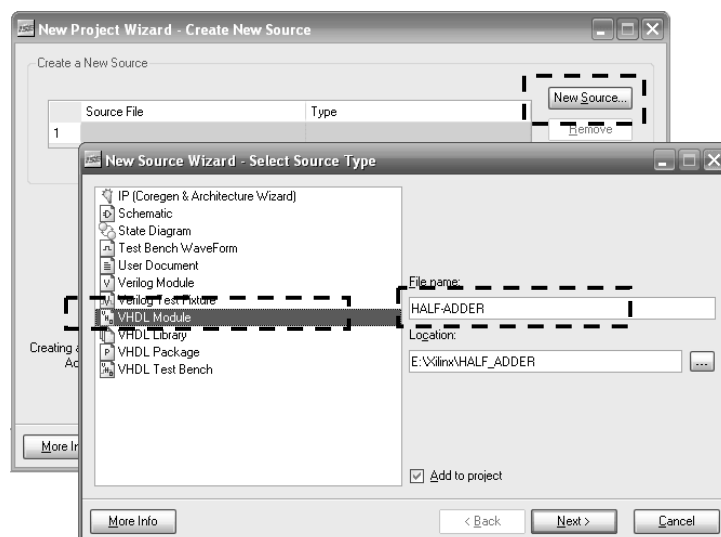
การทดลอง

วงจร Half-Adder

1. ออกแบบวงจรในรูปแบบที่ 1.1 ด้วย ISE WebPACK เริ่มที่เมนู Start → Programs → Xilinx ISE 8.1i → Project Navigator หรือ ดับเบิลคลิกที่  จะได้นหน้าต่างโปรแกรม Xilinx-ISE
2. สร้าง Project โดยเลือกเมนู File → New Project ที่หน้าต่าง New Project Wizard – Create New Project ให้ป้อนชื่อ Project ที่ช่อง Project Name ว่า HALF_ADDER และที่ช่อง Top-Level Source Type เป็นรูปแบบ HDL จากนั้นคลิก Next ไปที่หน้าต่าง New Project Wizard – Device Properties ตั้งค่าของ Synthesis Tool และ Simulator เป็น XST และ ISE Simulator แล้วคลิก Next



3. ที่หน้าต่าง New Project Wizard – Create New Source จุดนี้เป็นการเพิ่มชิ้นงานใน Project ให้เลือก New Source... เลือกรูปแบบ Source เป็น VHDL module ซึ่งเป็นรูปแบบไฟล์สร้างผังวงจรและตั้งชื่อไฟล์ว่า HALF_ADDER จากนั้น คลิก Next และ Finish จะพบว่า มี Source ที่สร้างไว้แล้ว ให้คลิก Next เพื่อไปที่หน้าต่างต่อไป



4. หน้าต่าง New Source Wizard – Define Module เป็นส่วนที่ใช้กำหนดอินพุต/เอาต์พุตใน Entity ให้ประกาศ ดังนี้

Port Name	Direction
a	in
b	in
sum	out
carry	out

หลังจากนั้นให้กด Next และ Finish ตามลำดับ

5. หน้าต่าง New Project Wizard – Add Existing Sources เป็นส่วนที่ใช้เพิ่ม Source ที่มีอยู่แล้ว ใช้ประโยชน์เมื่อวงจรที่ต้องการออกแบบมีส่วนที่ทำได้ทำไว้แล้ว สามารถที่จะเอาส่วนนั้นๆมาใช้ได้โดยไม่ต้องสร้างขึ้นมาใหม่ ในส่วนนี้ให้คลิก Next ไปที่หน้าต่าง New Project Wizard – Project Summary แล้วกด Finish



6. หลังจากนั้นจะพบหน้าต่างของโปรแกรมภาษา VHDL ซึ่ง Xilinx สร้างขึ้นมาให้ ให้ทำการเพิ่มการอธิบายการทำงานของ Half-Adder ในส่วน Architecture ด้วย

```
SUM    <= A xor B;
CARRY <= A and B;
```

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity HALF_ADDER is
    Port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          SUM : out STD_LOGIC;
          CARRY : out STD_LOGIC);
end HALF_ADDER;

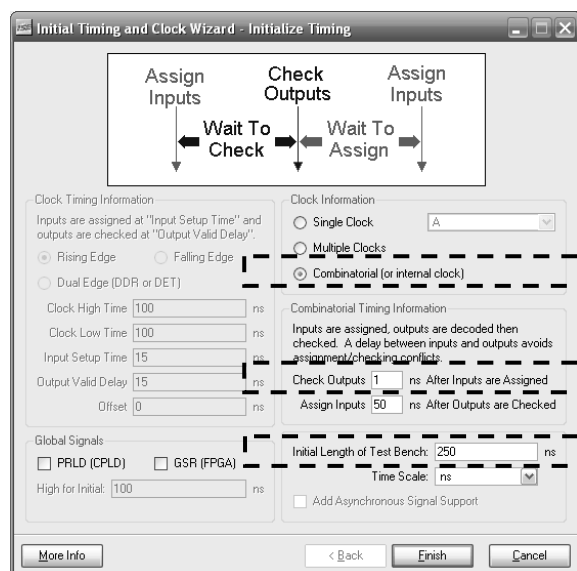
architecture Behavioral of HALF_ADDER is
begin
    SUM <= A xor B;
    CARRY <= A and B;
end Behavioral;

```

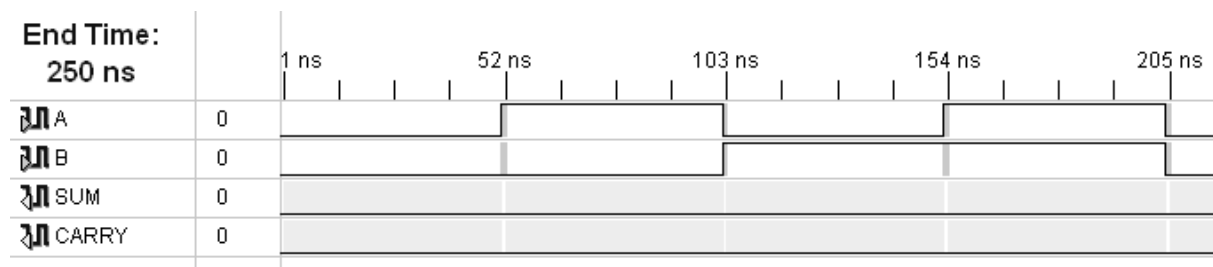
จากนั้นให้ทำการบันทึก

หมายเหตุ ทุกครั้งที่มีการเปลี่ยนแปลงแก้ไขโปรแกรม ให้ทำการบันทึกก่อนที่จะทำงานต่อไป

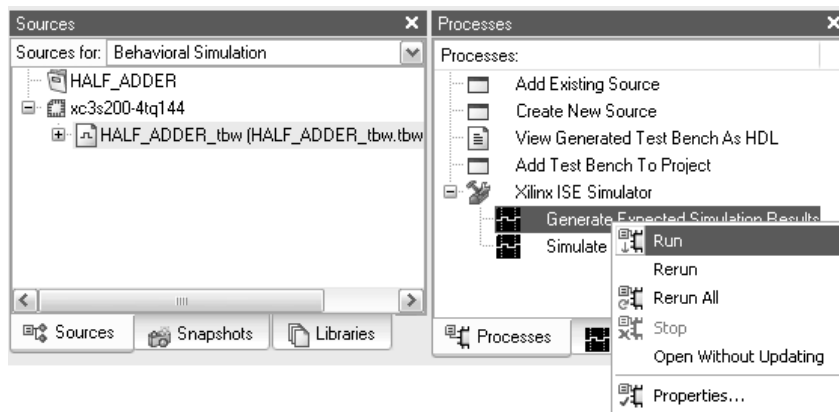
- สร้าง Test Bench WaveForm สำหรับจำลองการทำงาน โดยไปที่เมนู Project → New Source... ตั้งชื่อไฟล์ว่า *Half_Adder_tbw* เลือกรูปแบบเป็น *Test Bench WaveForm* แล้วคลิก Next ไปที่หน้าต่าง New Source Wizard – Associate Source แล้วเลือกวงจรที่ต้องการจำลองการทำงาน ในที่นี้คือ *Half_Adder* คลิก Next และ Finish ตามลำดับ จะปรากฏหน้าต่าง Initial Timing and Clock Wizard – Initialize Timing สำหรับการตั้งค่ารูปแบบการพิจารณาสัญญาณ ตั้งค่า Clock Information เป็น *Combinatorial (or internal clock)* เปลี่ยนช่วงเวลาในการ Check Outputs ใน Combinatorial Timing Information เป็น 1 ns และเปลี่ยนเวลาในการจำลองการทำงาน Initial Length of Test Bench เป็น 250 ns แล้วกด Finish จะได้ Timing Diagram ที่ใช้ในการจำลองการทำงาน



8. ปรับแต่งสัญญาณ Input ของวงจรเพื่อใช้ในการทดสอบ



9. จำลองการทำงานของวงจรด้วย ISE Simulator จาก Test Bench WaveForm โดยไปที่หน้าต่าง Sources เปลี่ยนรูปแบบเป็น Behavioral Simulation จากนั้นคลิกที่ไฟล์ Half_Adder_tbw เพื่อเลือกสัญญาณที่จะจำลองการทำงาน แล้วไปที่หน้าต่าง Processes ที่แท็บ Processes คลิกขวาที่ Xilinx ISE Simulator → Generate Expected Simulation Results คลิก Run เพื่อเริ่มทำการจำลองการทำงาน



10. เมื่อทำการวิเคราะห์เสร็จ จะมีไดอะล็อก Test Bench WaveForm Editor ถามเพื่อจะแสดงผล ให้ตอบ Yes และได้คลิก Expected Results ถามถึงการแสดงผลความแตกต่างจากผลการจำลองการทำงานเดิม ให้ตอบ Yes จะได้สัญญาณที่เกิดจากการจำลองการทำงานเกิดขึ้น

11. พิจารณาสัญญาณที่เกิดจากการจำลองการทำงานและบันทึกผล

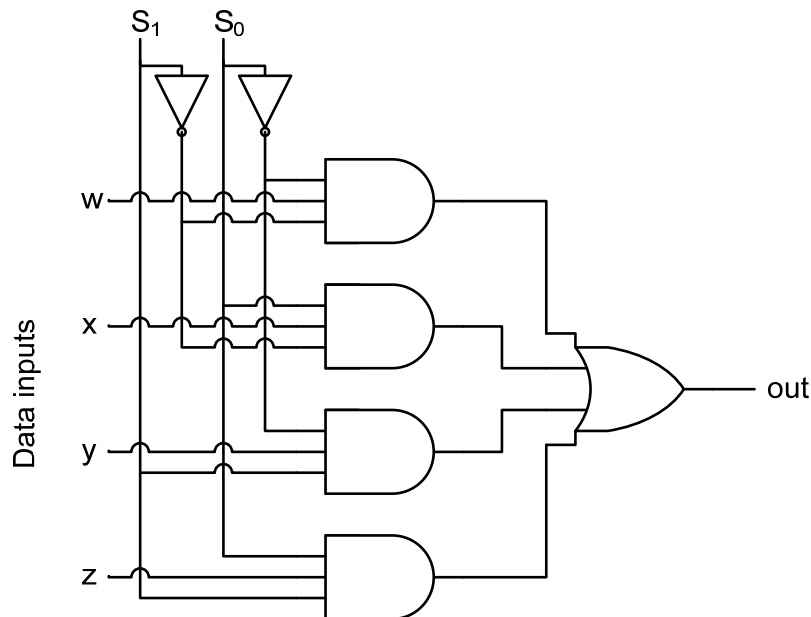
12. ศึกษาโปรแกรม Half-Adder ที่เขียนด้วย VHDL และอธิบายส่วนของโปรแกรม

ศึกษาและเปรียบเทียบการทำงานแบบ Sequential

1. ออกแบบโปรแกรมจากรูปที่ 1.3 และ 1.4
2. จำลองการทำงานของวงจรและเปรียบเทียบผลลัพธ์ที่ได้
3. ศึกษาและอธิบายส่วนของโปรแกรมที่เขียนขึ้น

วงจร 4-bit Multiplexer

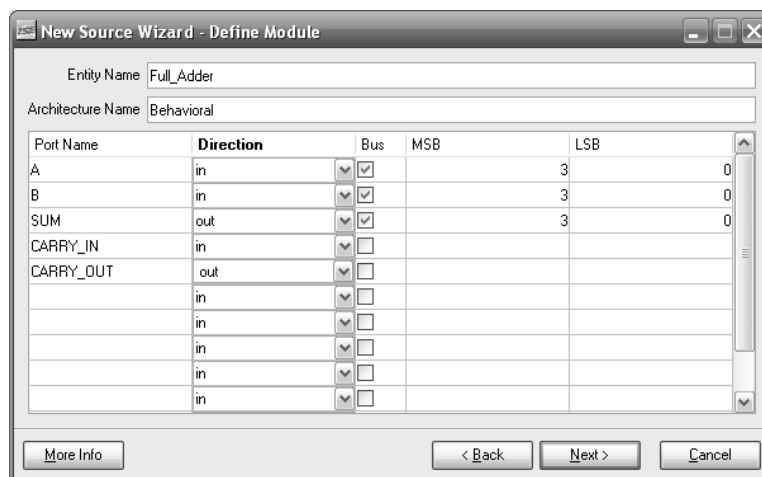
1. ออกแบบโปรแกรมวงจร 4-bit Multiplexer แบบให้มีการทำงานแบบ Sequential



2. จำลองการทำงานของวงจรและเปรียบเทียบผลลัพธ์ที่ได้
3. ศึกษาและอธิบายส่วนของโปรแกรมที่เขียนขึ้น

วงจร 4-bit Full-Adder

1. ออกแบบ วงจร 4-bit Full-Adder ด้วยภาษา VHDL โดยในขั้นตอน Define Module ที่เป็นการตั้งค่า ภายใน Entity ให้ตั้งค่าอินพุตและเอาท์พุต ดังรูป



หมายเหตุ: ในวงจรนี้ต้องใช้คำสั่งในการแปลงค่า std_logic_vector เป็น integer เพื่อใช้ในการคำนวณและ integer เป็น std_logic_vector เพื่อใช้ในการแสดงผล ให้ทำการค้นหาคำสั่งที่จะต้องใช้ใน Language Templates โดยไปที่เมนูบาร์ Edit -> Language Templates...

2. จำลองการทำงานของวงจรและบันทึกผล
3. ศึกษาและอธิบายส่วนของโปรแกรมที่เขียนขึ้น