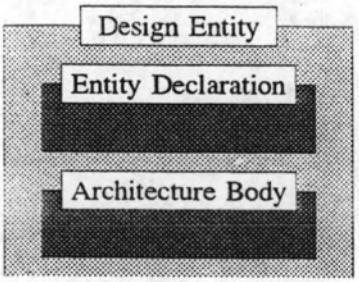
บทที่ 2

ภาษาวีเอชดีแอลเบื้องต้น

VHDL เป็นตัวย่อจากตัวย่อสองตัวคือ VHSIC กับ HDL ซึ่ง VHSIC ย่อมาจาก Very Hi-Speed IC และ HDL ย่อมาจาก Hardware Description Language เมื่อรวมความแล้วภาษาวีเอชดีแอลหมายถึง ภาษาที่ใช้ในการอธิบายฮาร์ดแวร์ที่มีความเร็วสูง

องค์ประกอบหลัก[1]

แบบจำลองภาษาวีเอชดีแอลใด ๆ ประกอบด้วยส่วนใหญ่ ๆ สองส่วนที่บรรจุอยู่ใน Design Entity คือ Entity Declaration และ Architecture Body ดังแสดงในรูปที่ 2-1



รูปที่ 2-1 ส่วนประกอบของแบบจำลองภาษาวีเอชดีแอล

[1] David R. Coelho, The VHDL Handbook, Kluwer Academic Publisher, 1989

1. Entity Declaration มีหน้าที่หลักในการกำหนดชื่อสัญญาณและทิศทางการติดต่อระหว่าง Design Entity กับโลกภายนอก โครงสร้างบางส่วนของการประกาศ Entity มีดังนี้

```
ENTITY identifier IS
        -- generic clause
        GENERIC ( generic_list );
        -- port clause
        PORT ( port_list );
END identifier ;
```

identifier คือชื่อของ Design Entity ส่วน generic clause จะเป็นการกำหนดพารามิเตอร์ต่าง ๆ เช่น ค่าเวลาประวิง (Delay Time) โดยที่จะมีการผ่านค่าที่แน่นอนเมื่อมีการเรียกใช้ Design Entity นั้นภายหลัง ดังนั้น Design Entity หนึ่ง ๆ จึงสามารถให้ผลการวิเคราะห์ที่ต่างกันตามแต่ค่าพารามิเตอร์ที่ให้ ยกตัวอย่างเช่น การเปลี่ยนแปลงตระกูลไอซีจากซีมอสไปเป็นทีทีแอลที่มีหน้าที่การทำงานเหมือนกัน หรือการเลือกช่วงเวลาเข้าถึง (Access Time) ของหน่วยความจำ ก็สามารถทำได้โดยการเปลี่ยนค่าเวลาประวิงให้ generic เท่านั้นโดยไม่ต้องแก้ไขภายในตัวแบบจำลองเลย

port clause เป็นการกำหนด ชื่อ ชนิด และ ทิศทางของสัญญาณต่าง ๆ ที่จะรับหรือส่งหรือทั้งรับทั้งส่งข้อมูลกับภายนอก

2. Architecture Body เป็นส่วนที่ใช้อธิบายความสัมพันธ์ระหว่างสัญญาณเข้าและสัญญาณออกที่ผ่านมาจาก Entity แต่ละ Design Entity สามารถมี Architecture Body ได้หลายอันตามแต่วิธีที่ใช้อธิบาย โครงสร้างของการเขียน Architecture Body มีดังนี้

```
ARCHITECTURE identifier OF entity_name IS
  architecture_declarative_part
BEGIN
  architecture_statement_part
END identifier;
```

identifier คือชื่อของ architecture และ entity_name ในที่นี้จะต้องเป็นชื่อเดียวกันกับ identifier ที่ประกาศไว้ใน Entity Declaration นั่นก็คือชื่อของ Design Entity นั่นเอง

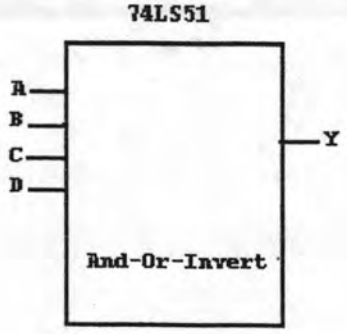
เปรียบเทียบภาษาวีแอชดีแอลกับการใช้แผนผังภาพ

เพื่อให้ผู้ที่คุ้นเคยกับการกับการเขียนวงจรโดยใช้แผนผังภาพ ได้มองภาพแบบจำลองภาษาวีเอชดีแอลได้ชัดขึ้น จึงขอเปรียบเทียบการอธิบายวงจรเชิงเลขด้วยภาษาวีเอชดีแอลกับการใช้แผนผังภาพดังตัวอย่างต่อไปนี้

1. Entity Declaration กับ Symbol

รูปที่ 2-2 แสดงการเปรียบเทียบการเขียน Entity Declaration ได้เท่ากับ การเขียน Symbol ของ 2-Wide-2-Input And-Or-Invert Gate ซึ่งสามารถให้ข้อมูลที่เหมือนกันดังนี้

1. จำนวนขาอินพุตของอุปกรณ์ ในตัวอย่างนี้มี 4 ขาได้แก่ ขา A,B,C และ D
2. จำนวนขาเอาท์พุตของอุปกรณ์ มี 1 ขาคือขา Y
3. หน้าที่ของอุปกรณ์ ในตัวอย่างนี้หน้าที่ของอุปกรณ์รู้ได้จากชื่อของอุปกรณ์ แต่สำหรับ เกตที่มีขนาดเล็ก จะสามารถรู้ได้จากรูปร่างของอุปกรณ์อีกด้วย

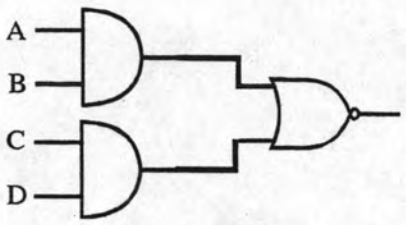



```
ENTITY And_Or_Invert IS
GENERIC (delay_time:time:=10ns);
PORT (A,B,C,D : IN std_ulogic;
                Y : OUT std_ulogic);
END And_Or_invert;
```

รูปที่ 2-2 แสดงการเปรียบเทียบ การการวาด Symbol กับการเขียน Entity

อย่างไรก็ดีวิธีใช้แผนผังภาพจะต้องทราบว่าต้องระบุใช้ไอซีตระกูลทีทีแอลเบอร์ 74LS51 ส่วนวิธีใช้ภาษาวีเอชดีแอลได้มีการประกาศ Generic ไว้ ขอให้ผู้อ่านลองนึกภาพวงจรที่มีขนาดใหญ่ๆ หากต้องการจะเปลี่ยนแปลงเทคโนโลยีการผลิต เช่น จากซีมอสเป็นทีทีแอล หรือหากต้องการรวมวงจรรวมขนาดเล็กหลาย ๆ ตัว ไว้ในวงจรรวมขนาดใหญ่เพียงตัวเดียว ถ้าออกแบบด้วยแผนผังภาพอาจจะต้องวาดวงจรใหม่ทั้งหมด เพราะแต่ละเทคโนโลยีใช้ Symbol ไม่เหมือนกัน แต่ถ้าใช้ภาษาวีเอชดีแอลเพียงแต่เปลี่ยนค่าพารามิเตอร์ให้ตรงกับเทคโนโลยี ก็จะสามารถนำแบบจำลองเดิมที่แก้ไขเพียงเล็กน้อยไปวิเคราะห์ หรือสังเคราะห์ได้อย่างถูกต้องแล้ว

2. Architecture Body กับ Schematic




```
ARCHITECTURE behav OF and_or_invert IS
BEGIN
        Y <= not ((A AND B) OR (C AND D));
END behav;
```

รูปที่ 2-3 แสดงการเปรียบเทียบการวาด Schematic กับการเขียนArchitecture Body

จากรูปที่ 2-3 แสดงการเปรียบเทียบการวาด Schematic กับการเขียน Architecture Body ซึ่งมีความหมายสมมูลกัน ในกรณีที่วงจรมีขนาดใหญ่จะเห็นได้ชัดว่าแบบจำลอง สื่อความหมายได้ดีกว่า ยกตัวอย่างเช่น แบบจำลองของ 10-bit Synchronous Counter มีแค่ 4-5 บรรทัด ดังนี้

```
PROCESS(clk)
BEGIN
        IF (rising_edge(clk)) THEN
                counter <= counter + "1";
        END IF;
END PROCESS;
```

ส่วนแผนผังภาพของวงจรนี้จะต้องใช้ประมาณ 30-40 เกต ทำให้ผู้อื่นอาจไม่เข้าใจหน้าที่การทำงานได้ดีเท่ากับการใช้ภาษาวีเอชดีแอล

รูปแบบการอธิบาย

การเขียนแบบจำลองในหัวข้อที่แล้วเป็นเพียงหนึ่งในวิธีการอธิบายเท่านั้น แท้ที่จริงแล้วรูปแบบการอธิบายแบบจำลองภาษาวีเอชดีแอลมีอยู่ด้วยกัน 3 วิธี

1. การอธิบายเชิงพฤติกรรม (Behavioral Description) เป็นการอธิบายวงจรที่บอกว่าสัญญาณแต่ละสัญญาณมีความสัมพันธ์กันอย่างไร แต่ไม่ได้บอกว่าทำได้โดยวิธีใดใช้เกตอะไร

การอธิบายแบบนี้จะทำให้ผู้อ่านเข้าใจหน้าที่การทำงานของชิ้นงานได้โดยง่าย และผู้ออกแบบสามารถวิเคราะห์หาที่ผิดได้ง่ายกว่าวิธีอื่น

2. การอธิบายการเชิงกระแสข้อมูล (Data-flow Description) เป็นการอธิบายว่าสัญญาณขาเข้ามีการเคลื่อนที่ไปยังสัญญาณขาออกด้วยการผ่านขั้นตอนอย่างไรบ้าง การอธิบายแบบนี้ผู้อ่านจะเห็นการเคลื่อนที่ของข้อมูล และผู้ออกแบบสามารถนำแบบจำลองแบบนี้ไปวิเคราะห์ด้วยการใส่ชุดสัญญาณทดสอบ (Test Vector) ชุดเดียวกับวิธีแรกอีกทั้งยังสามารถนำแบบจำลองแบบนี้ไปสังเคราะห์สร้างแผนผังภาพ ได้อีกด้วย

3. การอธิบายเชิงโครงสร้าง (Structural Description) เป็นการอธิบายวงจรในขั้นต่ำ โดยการอธิบายการต่อเชื่อมของอุปกรณ์ภายใน มีการอธิบายคล้ายกันกับการต่อเชื่อมของการใช้แผนผังภาพซึ่งจะเหมือนกับวงจรที่เกิดขึ้นจริง ทำให้สามารถวิเคราะห์หาความผิดพลาดเนื่องจากเวลาหน่วงของสัญญาณที่ผ่านเกตต่าง ๆ ได้ดีที่สุด

การที่มีการอธิบายแบบจำลองต่าง ๆ กันถึงสามวิธีก็เพราะแต่ละวิธีมีข้อดีข้อเสียต่าง ๆ กัน หากผู้ออกแบบทำตามขั้นตอนแล้ว ควรเขียนแบบจำลองเชิงพฤติกรรมเป็นอันดับแรกทั้งนี้จะทำให้วิเคราะห์และทำการแก้ไขได้ง่าย เมื่อได้ผลการวิเคราะห์เป็นที่น่าพอใจแล้วจะสามารถเก็บชุดสัญญาณทดสอบไปใช้กับแบบจำลองแบบอื่น ทั้งนี้เพื่อเปรียบเทียบผลลัพธ์ที่ถูกต้องที่ได้จากแบบจำลองชนิดนี้กับผลลัพธ์ที่ได้จากแบบจำลองชนิดอื่น

ในขั้นต่อมาควรเขียนแบบจำลองเชิงกระแสข้อมูล เพราะว่าแบบจำลองแบบนี้มีลักษณะคล้ายกับการเขียนแบบจำลองแบบแรกแต่ให้ข้อมูลมากกว่าคือ ข้อมูลแต่ละจุดเกิดจากสัญญาณใดกระทำอย่างไรกับสัญญาณใดบ้าง ทำให้ผู้ออกแบบทราบการเคลื่อนของข้อมูลมาจะต้องผ่านเกตใดหรือวงจรใด อันจะนำไปสู่การเขียนแบบจำลองในขั้นสุดท้ายคือแบบเชิงโครงสร้าง ซึ่งจะให้ข้อมูลทางเวลาและการต่อเชื่อมของเกตต่าง ๆ ที่สามารถนำไปสร้างวงจรได้จริง

ตัวอย่างการเขียนแบบจำลองทั้งสามแบบ แสดงได้ดังโปรแกรมข้างล่างนี้

```
ENTITY shifter IS          -- entity declaration
      PORT ( shiftin        : IN    bit_vector (0 TO 3);      -- port clause
             shiftout       : OUT   bit_vector (0 TO 3);
             shiftctrl      : IN    bit_vector (0 TO 1));
END shifter;
```

ตัวอย่างการเขียน Entity ของ Shifter ซึ่งมี shiftin เป็นสัญญาณอินพุตขนาดสี่บิต ควบคุมการทำงานด้วย shiftctrl ขนาดสองบิต และให้สัญญาณเอาท์พุตทาง shiftout ขนาดสี่บิต

```
ARCHITECTURE behavioral OF shifter IS                -- archictecture body
BEGIN
        f1: PROCESS (shiftin,shiftctrl)              -- process statement
                VARIABLE shifted : bit_vector(0 TO 3);      -- process declaration part
        BEGIN                                        -- process statement part
                CASE shftctrl IS
                WHEN "00" => shifted := shiftin;    -- don't shift
                WHEN "01" => shifted := shiftin(1 TO 3) & '0'; -- shift left
                WHEN "10" => shifted := '0' & shiftin(0 TO 2);  -- shift right
                WHEN "11" => shifted := shiftin(0) & shiftin(0 TO 2); --Arith. shift right
                END CASE
                shiftout <= shifted AFTER 10ns;
        END PROCESS f1;
END behavioral;
```

จะเห็นได้ว่าแบบจำลองเชิงพฤติกรรมข้างบนนี้เป็นแบบที่เข้าใจง่าย ผู้อ่านจะทราบหน้าที่ของแบบจำลองว่ามีหน้าที่ในการเลื่อนข้อมูลใน shiftin แล้วให้ผลลัพธ์ทาง shiftout ซึ่งการเลื่อนข้อมูลแบบใดนั้นขึ้นอยู่กับสัญญาณอินพุตอีกอันหนึ่งคือ shiftctrl

```
ARCHITECTURE data_flow OF shifter IS
BEGIN
        shiftout(3) <=  shiftin(3) AFTER 10ns WHEN shiftctrl = "00" ELSE
                        '0'        AFTER 10ns WHEN shiftctrl = "01" ELSE
                        shiftin(2) AFTER 10ns WHEN shiftctrl = "10" ELSE
                        shiftin(2) AFTER 10ns;
        shiftout(2) <=  shiftin(2) AFTER 10ns WHEN shiftctrl = "00" ELSE
                        shiftin(3) AFTER 10ns WHEN shiftctrl = "01" ELSE
                        shiftin(1) AFTER 10ns WHEN shiftctrl = "10" ELSE
                        shiftin(1) AFTER 10ns;
        shiftout(1) <=  shiftin(1) AFTER 10ns WHEN shiftctrl = "00" ELSE
                        shiftin(2) AFTER 10ns WHEN shiftctrl = "01" ELSE
                        shiftin(0) AFTER 10ns WHEN shiftctrl = "10" ELSE
                        shiftin(0) AFTER 10ns;
```

```
    shiftout(0) <=    shiftin(0) AFTER 10ns WHEN shiftctrl = "00" ELSE
                      shiftin(1) AFTER 10ns WHEN shiftctrl = "01" ELSE
                      '0'        AFTER 10ns WHEN shiftctrl = "10" ELSE
                      shiftin(0) AFTER 10ns;
END data_flow;
```

แบบจำลองเชิงกระแสข้อมูลข้างบนนี้ จะอธิบายว่าสัญญาณเอาท์พุตแต่ละตัว เกิดขึ้นได้อย่างไร และเป็นผลของสัญญาณอินพุตอะไรบ้าง

```
ARCHITECTURE structural OF shifter IS
    COMPONENT mux                    -- architecture declaration part
        PORT(a,b,c,d,: IN bit;sel0,sel1: IN bit; y: OUT bit);
    END COMPONENT;
    FOR u1,u2,u3,u4  : mux   USE ENTITY WORK.mux4x1(behav);

BEGIN
    u1:mux  PORT MAP (shiftin(3),      '0',shiftin(2),shiftin(2),shiftctrl(0),shiftctrl(1),shiftout(3));
    u2:mux  PORT MAP (shiftin(2),shiftin(3),shiftin(1),shiftin(1),shiftctrl(0),shiftctrl(1),shiftout(2));
    u3:mux  PORT MAP (shiftin(1),shiftin(2),shiftin(0),shiftin(0),shiftctrl(0),shiftctrl(1),shiftout(1));
    u4:mux  PORT MAP (shiftin(0),shiftin(1),      '0',shiftin(0),shiftctrl(0),shiftctrl(1),shiftout(0));
END structural;
```

จากผลของแบบจำลองแบบเชิงกระแสข้อมูล จะทำให้สามารถเขียนแบบจำลองในแบบสุดท้ายคือแบบเชิงโครงสร้างได้ว่าเกิดจาก multiplexer ที่มีสี่อินพุตและการเชื่อมต่อสายสัญญาณตามที่ได้เขียนไว้ในแบบจำลองเชิงกระแสข้อมูล

อย่างไรก็ตามในปัจจุบันมีเครื่องมือที่ทันสมัยและมีความสามารถมากยิ่งขึ้น ซึ่งสามารถสังเคราะห์วงจรโดยการสร้างแผนผังภาพ จากแบบจำลองแบบเชิงกระแสข้อมูล หรือบางส่วนจากแบบจำลองเชิงพฤติกรรมโดยอัตโนมัติ เป็นเหตุให้ผู้ออกแบบสมัยนี้มักเขียนแบบจำลองที่ง่ายต่อการเข้าใจในและสามารถสังเคราะห์ได้ในคราวเดียวกัน ทำให้ยากแก่การแยกแยะว่าเป็นแบบจำลองชนิดเชิงพฤติกรรมหรือเชิงกระแสข้อมูลกันแน่ ส่วนการอธิบายแบบเชิงโครงสร้างมักไม่นิยมใช้แล้ว เพราะเมื่อสังเคราห์ได้แผนภาพแล้ว ก็จะไม่จำเป็นที่จะต้องเขียนแบบจำลองเชิงโครงสร้างอีก

ต่อไป หรือหากต้องการแบบจำลองแบบนี้ ก็มีเครื่องมือสร้างแบบจำลองแบบเชิงโครงสร้างจากแผนภาพที่สังเคราะห์ขึ้นมาได้โดยอัตโนมัติ