

ใบงานที่ 4

เรื่อง การลองวิเคราะห์และประเมินผลคณิตศาสตร์และตรรกศาสตร์

จดทำโดย

นาย นิวัฒน์ บุญเรืองต้นพงษา

รหัส 65543206016-9

ชั้น ENGCE200\_SEC1

สูญเสีย

อาจารย์ สุวนิภา ลุณณะ

เขียนฝึกหัดนี้ เนื่องต่อหน้าห้อง บนวิชา

ENGEE200

การออกแบบระบบดิจิทัล

(Digital System Design)

หลักสูตร วศ.บ. วิศวกรรมคอมพิวเตอร์

สาขาวิชา วิศวกรรมไฟฟ้า คณะ วิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2566

# ຈຸດປະສົງຄໍາການສອນ

- ປັບຕິກາດລອງຈວາທີ່ຢູ່ປະມານຜລຄນິຕຄສຕ້ວແລະຕວກີກາຍ
- ອອກແບ່ງຈວາທີ່ຢູ່ປະມານຜລຄນິຕຄສຕ້ວແລະຕວກີກາຍ

## 1.) ການເພີ້ມ

### • ວິຊາການເພີ້ມຄວາມ (Binary Adder)

ເຮົາສົນໃຈປ່ຽນຈຳອາໄດ້ເປັນ 2 ແບບ Half Adder ແລະ Full Adder

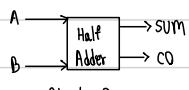
#### ການ Half Adder

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 10 = 0 + \text{carry of } 1 \text{ info next position}$$



A	B	SUM	CO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth Table

#### ການ Full Adder



Block Diagram

Ci	A	B	SUM	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Truth Table

### • ວິຊີ່ອອກແບ່ງຈຳ (binary subtractor)

#### ການ Half Subtractor

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 = 1 \quad \text{ຕີເຫັນທີ່ມີກຳນົດກຳນົດ 1}$$



A	B	SUM	Bo
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth Table

#### ການ Full Subtractor



Block Diagram

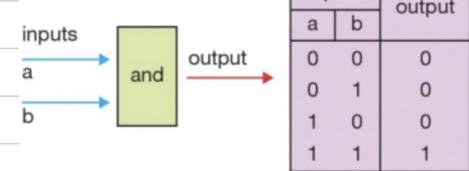
Bi	A	B	SUB	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Truth Table

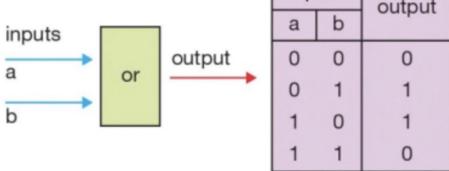
## ວິຊີ່ອອກແບ່ງຈຳຕະຫຼາກວິກາຍ

### Logic circuits

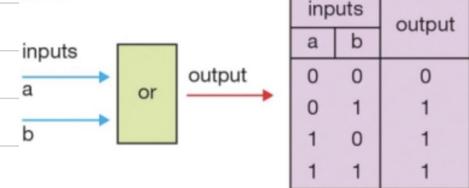
#### AND



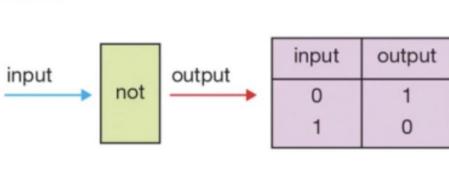
#### EXCLUSIVE OR



#### OR



#### NOT

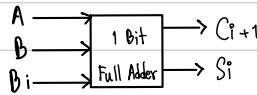


## 2.) บวกเลขฐาน 2

### 5 Bit Adder

**Step 1 : Problem Statement / Specification :** บวกเลขฐาน 2 ด้วย 5 bit Adder และพิสูจน์การทำงานของมัน

โครงสร้าง Cascade Adder หาๆ Adder 1 bit  $A_i, B_i$  ร่วมกับตัวนำเข้า  $C_i$  แล้วได้ตัวนำออกเป็น  $S_i$  และตัวนำต่อไป  $C_{i+1}$  ตาม 5 หลัก



**Step 2 : Conceptualization :** นำรายละเอียดการทำงานของรวมเขียนเป็น function table หรือ truth table

Input			Output	
$C_i$	$A_i$	$B_i$	$C_{i+1}$	$S_i$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

**Step 3 : Solution / Simplification**

จาก truth table สามารถเขียนเป็น function ของ FaceTime และ  $C_{i+1}$  ให้จาก Boolean Equation ได้ดังนี้

$S_i$	$C_i$	$A_i$	$B_i$
0	00	0	0
0	01	0	1
1	10	1	0
1	11	0	1

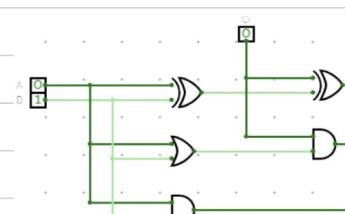
$$\begin{aligned}
 S_i &= C_i \bar{A}_i \bar{B}_i + \bar{C}_i \bar{A}_i B_i + \bar{C}_i A_i \bar{B}_i + C_i A_i B_i \\
 &= C_i (\bar{A}_i \bar{B}_i + A_i B_i) + \bar{C}_i (\bar{A}_i \bar{B}_i + A_i \bar{B}_i) \\
 &= C_i (\bar{A}_i \oplus B_i) + \bar{C}_i (A_i \oplus B_i) \\
 &= C_i \oplus (A_i \oplus B_i)
 \end{aligned}$$

$S_i$	$C_i$	$A_i$	$B_i$
0	00	0	0
0	01	0	1
1	11	1	0
1	10	0	1

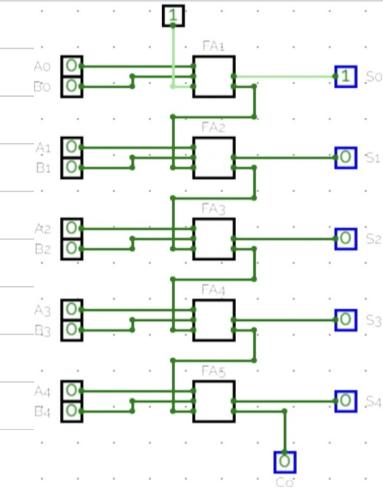
$$\begin{aligned}
 C_{i+1} &= A_i B_i + C_i B_i + C_i A_i \\
 &= A_i B_i + C_i (A_i + B_i) \\
 &= A_i B_i + C_i \oplus (A_i \oplus B_i)
 \end{aligned}$$

**Step 4 : Realization :** เนื่องจากผู้ที่ได้จากผู้ที่ได้ดูจะต้องใช้ 1 bit กับตัว 5 หาๆ 1 bit Full Adder

หาๆ 1 Bit Adder



หาๆ 5 Bit Adder



## 5 Bit Adder / Subtractor

**Step 1 : Problem Statement / Specification :** บอกรายละเอียดของวงจรที่ต้องประกอบด้วย Input , Output และฟังก์ชันการคำนวณของวงจรออกแบบจะ 5 Bit Adder/Subtractor ด้วยการ Combination ด้วย W คือ Control I/P และ Y คือ Data I/P ส่วน F คือ เส้นทางผลลัพธ์

**Step 2 : Conceptualization :** นำรายละเอียดการคำนวณของวงจรมาเขียนเป็น function table หรือ truth table

ถ้า  $w=0$ , Data I/P  $y$  จะเป็น O/P

ถ้า  $w=1$ , Data I/P  $y$  จะเป็น Complement

สรุปเป็น Selective Complement Truth Table ดังนี้

Input		Output
Control	Data	F
0	0	0
0	1	1
1	0	1
1	1	0

} Pass

} Complement

**Step 3 : Solution / Simplification**

จาก Truth Table สรุปผลลัพธ์เมื่อตั้งค่า W คือ

$$F = \bar{w}y + w\bar{y}$$

$$= w \oplus y$$

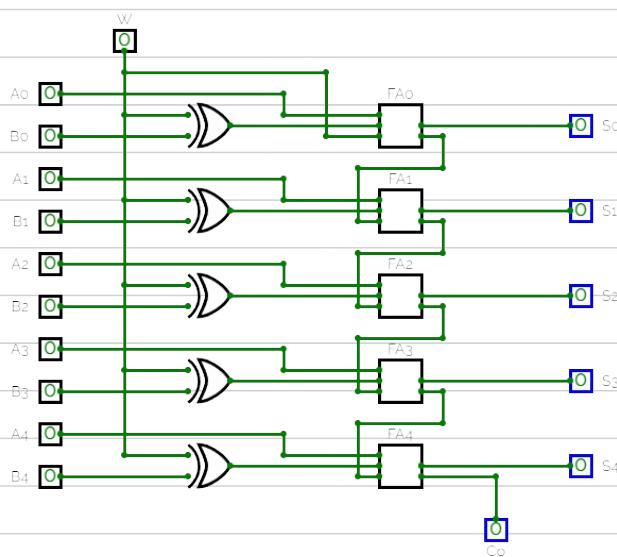
ดังนั้นใช้สมการคำนวณ Selective Complementation ได้โดยใช้ XOR

**Step 4 : Realization :** เขียนวงจรภาคฟังก์ชันได้

จากฟังก์ชันได้สมการดังนี้ 5 Bit Adder / Subtractor

$w = 0$  เป็นการซับtract และจะถูก Pass

$w = 1$  จะถูก Invert ทุกบิต เพื่อกำ 1's Complement และ 1 จาก  $w$  จะถูกบวกเข้าไปเก็บ Ci เพื่อกำ 2's Complement



## True / Complement / Zero / One

**Step 1 : Problem Statement / Specification :** ບອກຮາຍລະເອີຍຄວນວ່າຈະກຳຕ້ອງປະກອບດ້ວຍ Input , Output ແລະ ຜິດກຳທັນກາງກຳດຳກຳການຂອງຈະກຳ

- : ອອກແບບຜິດກຳທັນຂອງ True / Complement / Zero / One ທີ່ມີ TCO1 ຈະເອົາໄວ້ສຳກັນໃປປະກຳ Adder / Subtractor
- : TCO1 ສືບຕີ I/P Y ດ້ວຍເປີຕີ O/P F ດ້ວຍເປີຕີແລະນີ້ສັບມຸດຕາງປຸມ 2 ເສັ້ນ ( E = enable , W = Complement )

**Step 2 : Conceptualization :** ນໍາຮາຍລະເອີຍດາກກຳດຳກຳການຂອງຈະກຳເປົ້າເປັນ Function Table ທີ່ມີ F ຈະເປັນຄ່າ True ທີ່ມີ Complement ທີ່ມີ Zero ທີ່ມີ One ເຮັດວຽກ E , W ເປັນຕົວປຸມ

Control		Output
W	E	F
0	0	0
0	1	Y
1	0	1s
1	1	Y

ຈະອອກແບບ Enable Control ໄດ້ໂດຍໃຫ້ And Gate ຈຶ່ງ E ຈະເປັນຕົ້ງ Enable ຖ້າຈະຫຼື Y ດ້ວຍໄປໄດ້ຮູ້ຈົນແລະເອົາກຸດຈະຊຸກນຳນາຄອບຄຸມໂດຍ W ອີກຄວ້າ

E	Y	G = E · Y	O/P
0	0	0	0
0	1	0	0
1	0	0	Y
1	1	1	Y

## Step 3 : Solution / Simplification

ຈາກ Truth Table ຫຼັກນຳ 2 ຕາງໝົດກວ່ານີ້

W	E	Y	G = E · Y	F = W · (E · Y)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

ຖືກ E = 0 , Disable Data , G = 0

ເນື້ອ W = 0 , G = 0 ຈະ Pass ເທົ່ານີ້ F

W = 1 , G ຈະຊຸກ Complement ເທົ່ານີ້ F

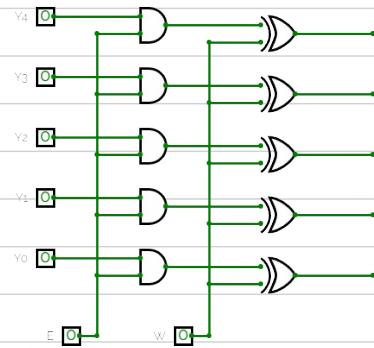
ຖືກ E = 1 , Disable Data , G = Y

ເນື້ອ W = 0 , Enable Data (EY) ຈະ Pass ເທົ່ານີ້ F

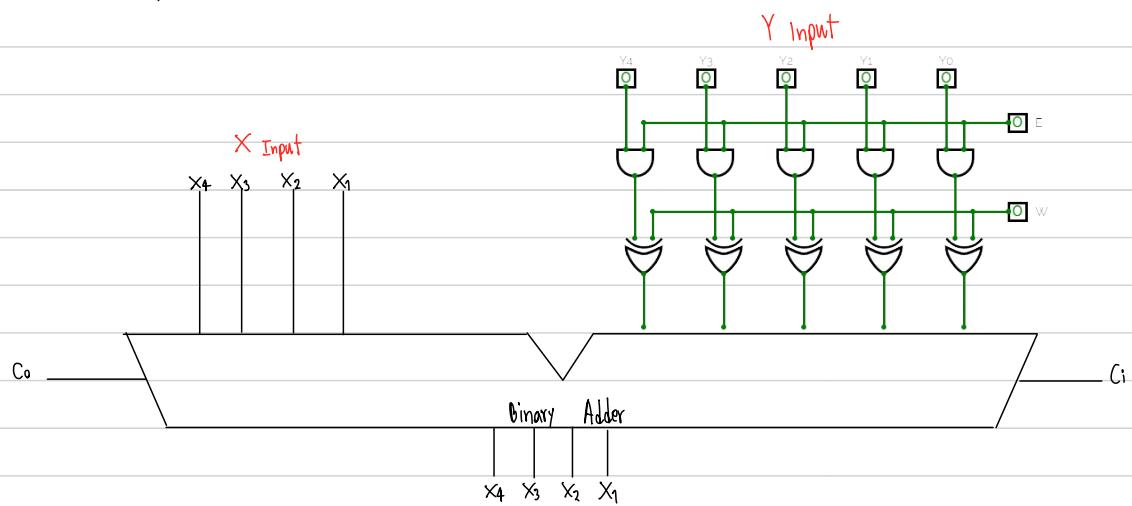
W = 1 , Enable ( EY ) ຈະຊຸກ Complement ເທົ່ານີ້ F

Step 4 : Realization : เนื่องจากพื้นที่จำกัด

Function ตาม Truth Table สามารถสร้างขึ้นโดยใช้ Gate ทำให้ Enable และ Control



เมื่อนำวงจร Tc01 มาต่อคอมบินেชัน Binary Adder จะได้ Au ที่สามารถคำนวณได้ตามพื้นที่ข้างต้น



## ອອກແບບ Second Binary Arithmetic Unit

**Step 1 : Problem Statement / Specification :** ບອກຮາຍລະເອີ້ນເດັບອອງຈະຈຳຕ້ອງປະກອບດ້ວຍ Input , Output ແລະ ຜົກໜັກການກຳທຳການນັບອອງຈະ

### ຕາມກຳ Function of AU

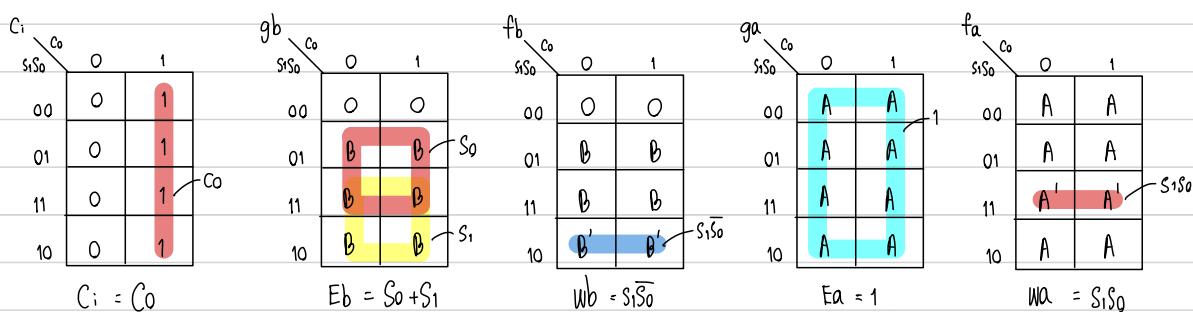
OP Code				Function	Operation	
M	S1	S0	Co			
0	0	0	0	A	Transfer A	
0	0	0	1	A+1	Increment A by 1	
0	0	1	0	A+B	Add A and B	
0	0	1	1	A+B+1	Increment the sum of A and B by 1	
0	1	0	0	A+B'	A plus one's complement of B	
0	1	0	1	A-B	Subtract B from A	
0	1	1	0	A'+B	B plus one's complement of A	
0	1	1	1	B-A	B minus A (or A'+B+1)	

**Step 2 : Conceptualization :** ນໍາຮາຍລະເອີ້ນເດັບການກຳທຳການນັບອອງຈະຈຳເປົ້າເປັນ Function Table ທີ່ອ Truth Table

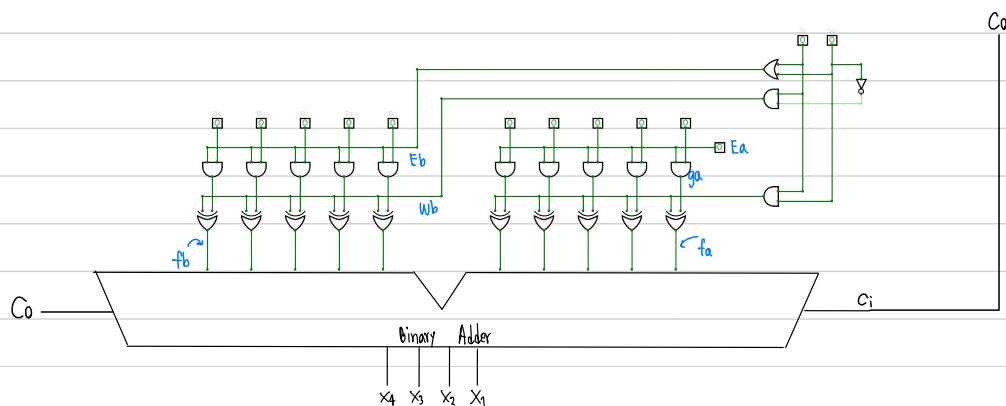
ຕາມການກຳທຳການນັບອອງຈະ TCO1

OP Code				Output					Function	Operation
M	S1	S0	Co	Ci	gb	fb	ga	fa		
0	0	0	0	0	0	0	A	A	A	Transfer A
0	0	0	1	1	0	0	A	A	A+1	Increment A by 1
0	0	1	0	0	B	B	A	A	A+B	Add A and B
0	0	1	1	1	B	B	A	A	A+B+1	Increment the sum of A and B by 1
0	1	0	0	0	B	B'	A	A	A+B'	A plus one's complement of B
0	1	0	1	1	B	B'	A	A	A-B	Subtract B from A
0	1	1	0	0	B	B	A	A'	A'+B	B plus one's complement of A
0	1	1	1	1	B	B	A	A'	B-A	B minus A (or A'+B+1)

**Step 3 : Solution / Simplification**

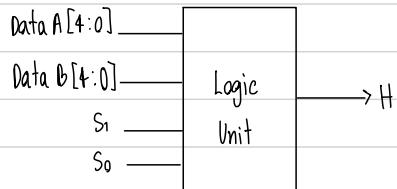


**Step 4 : Realization :** ເນັ້ນຈາກຜົກໜັກທີ່ໄດ້



## Logic Unit

**Step 1 : Problem Statement / Specification :** บอกรายละเอียดของวงจรที่ต้องประกอบด้วย Input , Output และฟังก์ชันการทำงานของวงจร ออกแบบ Logic Unit จะมี Input คือ Data A [ 4 : 0 ] และ B [ 4 : 0 ] จากนั้นเมื่อ Operation Select คือ S1 กับ S0 ส่ง入 Output

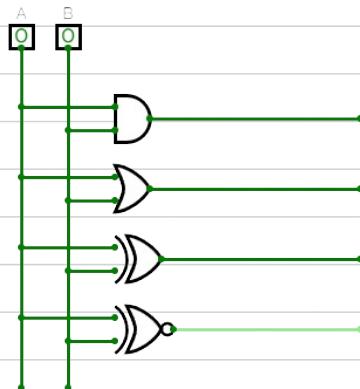


**Step 2 : Conceptualization :** นำรายละเอียดการทำงานของวงจรมาเขียนเป็น Function Table หรือ Truth Table

M	S1	S0	Co	Function	Operation
1	0	0	x	AiBi	AND
1	0	1	x	Ai+Bi	OR
1	1	0	x	Ai + Bi	XOR
1	1	1	x	Ai + Bi	XNOR

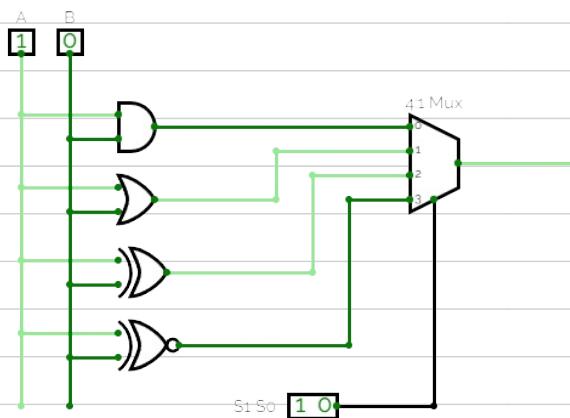
**Step 3 : Solution / Simplification**

จากตาราง Function ใช้ AND , OR , XOR , XNOR นำมาระบบแต่ละ Gate จะมี 5 ตัว



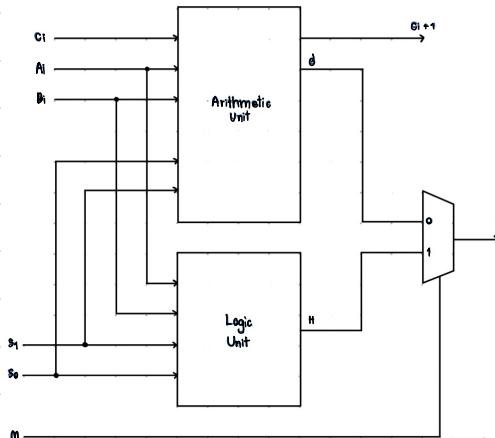
**Step 4 : Realization :** เขียนวงจรจากฟังก์ชันที่ได้

กำหนดให้ได้ 4 : 1 Multiplex โดยเมื่อ S1 กับ S0 เป็นขา Select ครบถ้วน 4 : 1 Multiplex จะมีขาข้อมูล 5 ตัว



## Arithmetic Unit และ Logic Unit (ALU)

**Step 1 : Problem Statement / Specification :** บอกรายละเอียดของวงจรที่ต้องประกอบด้วย Input , Output และฟังก์ชันการคำนวณของวงจร เมื่อออกแบบวงจร Logic Unit เรียบร้อยแล้วให้ทำ Output ของ 5 Bit ALU และ Output ของ Logic Unit ไปต่อเข้ากันเป็น 2 : 1 (Mux) ก็จะมีผล 5 ตัว



**Step 2 : Conceptualization :** นำรายละเอียดการทำงานของวงจรมาเป็น Function Table หรือ Truth Table

Truth Table ของ ALU

OP Code				Output					Function	Operation
M	S1	S0	Co	Ci	gb	fb	ga	fa		
0	0	0	0	0	0	0	A	A	A	Transfer A
0	0	0	1	1	0	0	A	A	A+1	Increment A by 1
0	0	1	0	0	B	B	A	A	A+B	Add A and B
0	0	1	1	1	B	B'	A	A	A+B+1	Increment the sum of A and B by 1
0	1	0	0	0	B	B'	A	A	A+B'	A plus one's complement of B
0	1	0	1	1	B	B'	A	A	A-B	Subtract B from A
0	1	1	0	0	B	B	A	A'	A'+B	B plus one's complement of A
0	1	1	1	1	B	B	A	A'	B-A	B minus A (or A'+B+1)

Truth Table ของ Logic Unit

M	S1	S0	Co	Function	Operation
1	0	0	x	AiBi	AND
1	0	1	x	Ai+Bi	OR
1	1	0	x	Ai + Bi	XOR
1	1	1	x	(Ai ⊕ Bi)	XNOR

### Step 3 : Solution / Simplification

Au

	$S_1 S_0$	$C_i$
$S_1 S_0$	00	0
	01	1
	11	1
	10	0

$$C_i = C_0$$

	$S_1 S_0$	$g_b$
$S_1 S_0$	00	0
	01	0
	11	B
	10	B'

$$F_b = S_0 \cdot S_1$$

	$S_1 S_0$	$f'_b$
$S_1 S_0$	00	0
	01	B
	11	B
	10	B'

$$W_b = S_1 \bar{S}_0$$

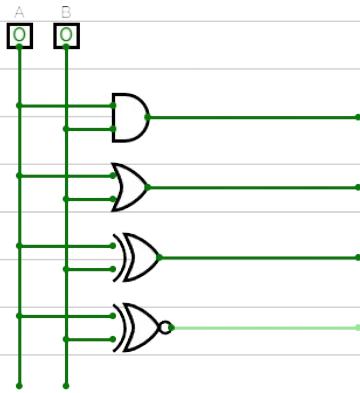
	$S_1 S_0$	$g_a$
$S_1 S_0$	00	0
	01	A
	11	A
	10	A

$$E_a = 1$$

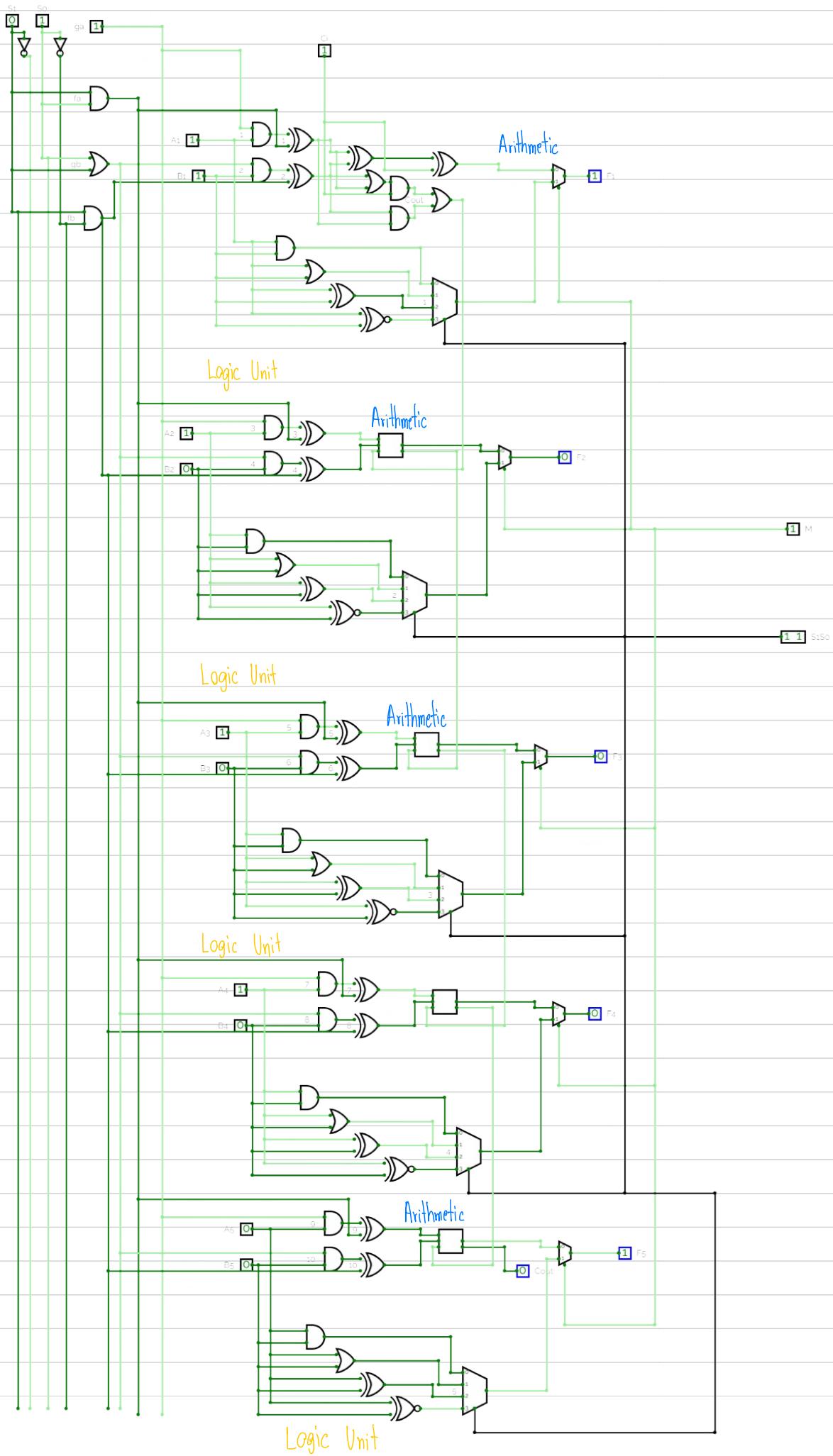
	$S_1 S_0$	$f_a$
$S_1 S_0$	00	0
	01	A
	11	A'
	10	A

$$W_a = S_1 S_0$$

Logic Unit



**Step 4 : Realization :** เน้นๆจะจากฟังก์ชันที่ได้



### 3) ເລກສາດລອງ

#### ໜີຈີ ALU

- ເປົ້າ VHDL ຂອງຈີ ALU

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

entity Alu is
  Port (
    A: in std_logic_vector(5 downto 1);
    B: in std_logic_vector(5 downto 1);
    S: in std_logic_vector(1 downto 0);
    M: in std_logic;
    ci: in std_logic;
    F: out std_logic_vector(5 downto 1)
  );
end entity Alu;

architecture Behavioral of Alu is
  signal and_gate : std_logic_vector(10 downto 1);
  signal Cout : std_logic_vector(5 downto 1);
  signal xor_gate : std_logic_vector(10 downto 1);
  signal mux_out : std_logic_vector(5 downto 1);
  signal ga: std_logic;
  signal gb: std_logic;
  signal fa: std_logic;
  signal fb: std_logic;
begin
  process(A, B, ci, S, M, and_gate, mux_out, ga, gb, fa, fb,
Cout, xor_gate)
  begin
    ga <= '1' ;
    gb <= S(1) or S(0) ;
    fa <= S(1)and S(0) ;
    fb <= (not S(0)) and S(1);

    --Bit 1
    and_gate(1) <= A(1) and ga ;
    and_gate(2) <= B(1) and gb;
    xor_gate(1) <= and_gate(1) xor fa;
    xor_gate(2) <= and_gate(2) xor fb;
    Cout(1) <= (ci and (xor_gate(1) or xor_gate(2))) or
(xor_gate(1) and xor_gate(2)) ;

    case S(1 downto 0) is
      when "00" => mux_out(1) <= A(1) and B(1);
      when "01" => mux_out(1) <= A(1) or B(1);
      when "10" => mux_out(1) <= A(1) xor B(1);
      when others => mux_out(1) <= A(1) xnor B(1);
    end case;
  end process;
end architecture;
```

case M is

```
when '0' => F(1) <= (xor_gate(1) xor xor_gate(2))xor ci;  
when others => F(1) <= mux_out(1);
```

end case;

--Bit 2

```
and_gate(3) <= A(2) and ga ;  
and_gate(4) <= B(2) and gb;  
xor_gate(3) <= and_gate(3) xor fa;  
xor_gate(4) <= and_gate(4) xor fb;
```

```
Cout(2) <= (Cout(1) and (xor_gate(3) or xor_gate(4))) or (xor_gate(3) and xor_gate(4)) ;
```

case S(1 downto 0) is

```
when "00" => mux_out(2) <= A(2) and B(2);  
when "01" => mux_out(2) <= A(2) or B(2);  
when "10" => mux_out(2) <= A(2) xor B(2);  
when others => mux_out(2) <= A(2) xnor B(2);
```

end case;

case M is

```
when '0' => F(2) <= (xor_gate(3) xor xor_gate(4))xor Cout(1);  
when others => F(2) <= mux_out(2);
```

end case;

--Bit 3

```
and_gate(5) <= A(3) and ga ;  
and_gate(6) <= B(3) and gb;  
xor_gate(5) <= and_gate(5) xor fa;  
xor_gate(6) <= and_gate(6) xor fb;
```

```
Cout(3) <= (Cout(2) and (xor_gate(5) or xor_gate(6))) or (xor_gate(5) and xor_gate(6)) ;
```

case S(1 downto 0) is

```
when "00" => mux_out(3) <= A(3) and B(3);  
when "01" => mux_out(3) <= A(3) or B(3);  
when "10" => mux_out(3) <= A(3) xor B(3);  
when others => mux_out(3) <= A(3) xnor B(3);
```

end case;

case M is

```
when '0' => F(3) <= (xor_gate(5) xor xor_gate(6))xor Cout(2);  
when others => F(3) <= mux_out(3);
```

end case;

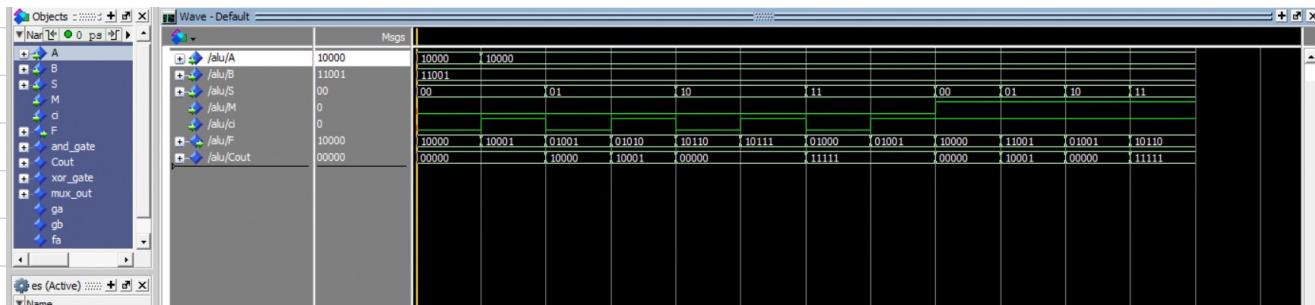
```

--Bit 4
    and_gate(7) <= A(4) and ga ;
    and_gate(8) <= B(4) and gb;
    xor_gate(7) <= and_gate(7) xor fa;
    xor_gate(8) <= and_gate(8) xor fb;
    Cout(4) <= (Cout(3) and (xor_gate(7) or xor_gate(8))) or (xor_gate(7) and xor_gate(8)) ;
case S(1 downto 0) is
    when "00" => mux_out(4) <= A(4) and B(4);
    when "01" => mux_out(4) <= A(4) or B(4);
    when "10" => mux_out(4) <= A(4) xor B(4);
    when others => mux_out(4) <= A(4) xnor B(4);
end case;
case M is
    when '0' => F(4) <= (xor_gate(7) xor xor_gate(8))xor Cout(3);
    when others => F(4) <= mux_out(4);
end case;

--Bit 5
    and_gate(9) <= A(5) and ga ;
    and_gate(10) <= B(5) and gb;
    xor_gate(9) <= and_gate(9) xor fa;
    xor_gate(10) <= and_gate(10) xor fb;
    Cout(5) <= (Cout(4) and (xor_gate(9) or xor_gate(10))) or (xor_gate(9) and xor_gate(10)) ;

case S(1 downto 0) is
    when "00" => mux_out(5) <= A(5) and B(5);
    when "01" => mux_out(5) <= A(5) or B(5);
    when "10" => mux_out(5) <= A(5) xor B(5);
    when others => mux_out(5) <= A(5) xnor B(5);
end case;
case M is
    when '0' => F(5) <= (xor_gate(9) xor xor_gate(10))xor Cout(4);
    when others => F(5) <= mux_out(5);
end case;
end process;
end Behavioral;
```

- Timing Diagram ALU



จากภาพดลลงบ่งชี้ว่า ALU ที่ได้ออกแบบปัจจุบันต้องตามเกณฑ์นี้

#### 4) สรุปผลการทดลอง

จะส. AU จากการทดลองนี้คือการประมวลผลคณิตศาสตร์และตรวจสอบbinary โดยใช้ Multiplex 2 : 1 ในการเลือกโดยมีความกว้าง AU กับ Logic Unit ส่วน S1 , SO , Carry In ใช้ในการเลือก Function การทำงาน จะส. AU นำไปใช้ประโยชน์ในการคำนวณของคณิตศาสตร์และการคำนวณตัวเลข เป็นต้น

#### 5) คำนวณหลักการทดลอง

##### 1. การออกแบบวงจรบีตติบบีต

ตอบ = ฝี 2 วิธี คือ

- Half Adder
- Full Adder

##### 2. บอกรหัสกการทำงานของบีตติบบีต

ตอบ = Complement สำลับสตางค์ของบีตติบบีต เช่น 0 -> 1 และ 1 -> 0 จะได้ 0010 -> 1101 และการคำนวณ Add One จะได้เป็น 1110

##### 3. คำนวณการออกแบบบีตติบบีตและบีตติบบีต 128 บิต จะต้องออกแบบอย่างไร

ตอบ = ใช้ Adder กับ AU และ Logic Unit จาก 5 Bit ให้ออกแบบเพิ่มจำนวนให้เป็น 128 Bit