Practical 5

MLLDYL002 (Dylan Muller)
LCHSAR003 (Sarvesh Luchmee)
SHMTAK004 (Tukudzwa Shumbamhuni)

Introduction

This practical entails the design of a 2-bit wide arithmetic logic unit (ALU) using discrete logic circuits.
The arithmetic logic unit (ALU) Fig 1. forms a critical component of the modern day central processing unit (CPU) and most commercially available microprocessors/micro-controllers.

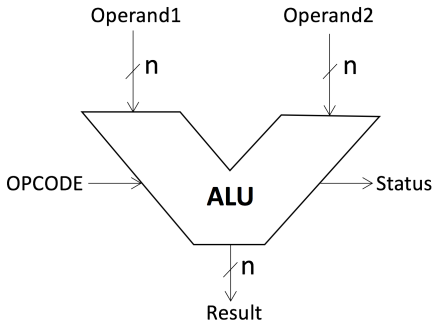

Fig 1. Arithmetic logic unit

It consists of three input registers/ports, two of which constitute the input data operands (1 and 2) and the third, an ALU operational code (opcode) operand/register. The opcode register determines the current arithmetic operation to be performed on operand 1 and operand 2 such as addition, subtraction, etc.

The ALU also contains an output register, withholding the result of the previous operation and a status register to provide additional information.

Design

A table of opcodes Fig 2. was provided in the practical 5 leaflet detailing the various functions required by the ALU for a given 3-bit opcode configuration. The design procedure thus consisted of designing the logic circuits for each function separately and then linking the various output bits using an 8-to-1 multiplexer. Restrictions were placed on the type of logic components required.

Opcode Bits			Function
F2	F1	F0	
0	0	0	A+B
0	0	1	A-B
0	1	0	A+1
0	1	1	A-1
1	0	0	A x 2
1	0	1	A / 2
1	1	0	A (AND) B
1	1	1	A (OR) B

Fig.2 Opcode table.

[A+B] Addition Logic Circuit

The addition of two bits is achieved through the use of half adder which may be extended to accommodate addition with multiple bits (full adder) Fig.3.
Since our data operands (A & B) are two bits wide, we require a half bit adder stage followed by a full bit adder Fig.4.

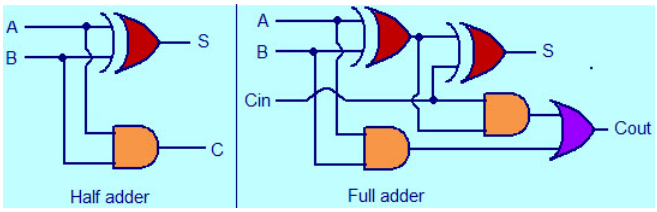

Fig.3 Half and Full adders.

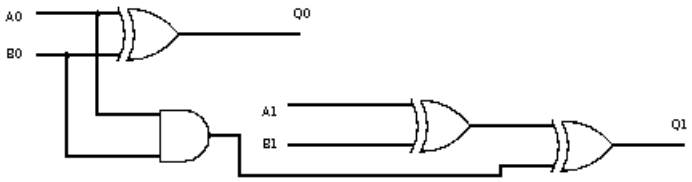

Fig.4 ALU adder implementation (without carry)

Addition of two operands (A and B) which are both two bits wide, involves the addition of the first bit of each operand (A0 +B0) [XOR], forming output bit 1 (Q0), followed by the addition of bits (A1+B1) which forms the second output bit (Q1).
Figure 4 is the resulting logic circuit implementation without an output carry.

[A+1] Increment Logic Circuit

Incremental addition of a register may be achieved using a configuration similar to that of Fig.4 with the operand register B replaced with (B0 = 1 [VCC]) and (B1 = 0 [GND]). Fig.5 depicts the resulting circuit without carry.

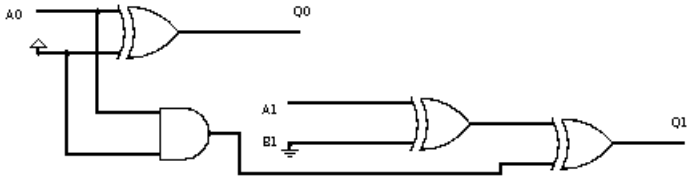

Fig.5 ALU incrementer implementation (without carry).

[A-B] Subtraction Logic Circuit

Subtraction of two bits may be realized through the use of a half subtractor Fig.6 or extended through the use of a full bit subtractor Fig.7.

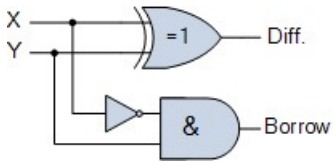

Fig.6 Half subtractor.

Operand A and B are both two bits wide, which, similar to our adder, requires a half bit adder stage followed by a full adder stage. The carry bit is suppressed and Fig.8 depicts the resulting implementatio

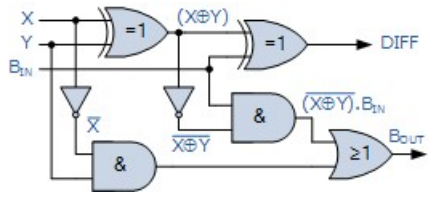

Fig.7 Full subtractor.

Bit 1 of A and B (A0 B0) are subtracted forming output bit Q0 followed by bit 2 of A and B (A1 B1) forming output bit Q1.

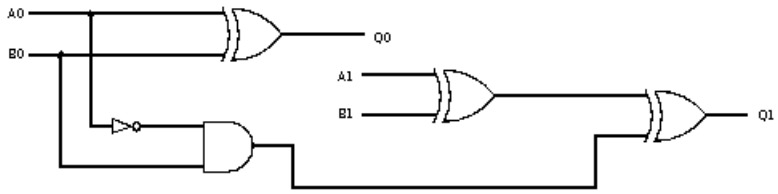

Fig.8 ALU subtractor implementation (without carry).

[A-1] Decrement Logic Circuit

Decremental subtraction of a register may be achieved using a configuration similar to that of Fig.8 with the operand register B replaced with (B0 = 1 [VCC]) and (B1 = 0 [GND]). Fig.9 depicts the resulting circuit without carry.

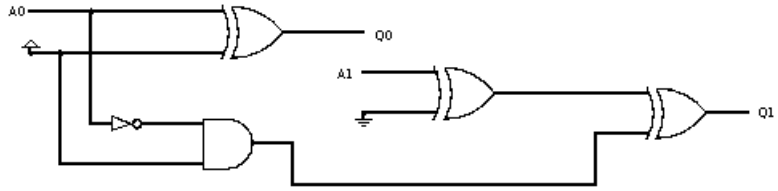

Fig.9 ALU decrementer implementation (without carry).

[Ax2] Multiply by 2

Multiplication by 2 within base 2 may be achieved through a bitwise left rotation/shift by 1. A LR (left rotate) shift register may be constructed using 2 multiplexers as shown in Fig 10.

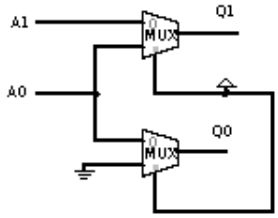

Fig.10 Left shift register.

Circuit operation depends on the MUX shift enable pin being connected to Vcc. A single left rotation follows, or multiplication by 2. The output bits are given by (Q1 and Q0).

[A/2] Divide by 2

Division by 2 within base 2 may be achieved through a bitwise right rotation/shift by 1. A RR (right rotate) shift register may be constructed using 2 multiplexers as shown in Fig 11.

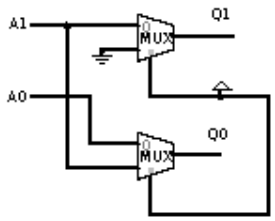

Fig.11 Left shift register.

Circuit operation depends on the MUX shift enable pin being connected to Vcc. A single right rotation follows, or division by 2. The output bits are given by (Q1 and Q0).

[A AND B] Bitwise AND

The ANDing of two bits may be achieved through the use of an AND gate. For the case of operands of two bit widths (A and B), ANDing is achieved through ANDing of bits 1 of A and B (A0,B0), forming output bit 1 (Q0) followed by bits (A1,B1), forming output bit 2 (Q1).
Figure 12. depicts the final circuit implementing bitwise AND for operand A and B.

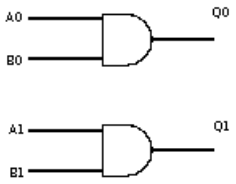

Fig.12 Bitwise AND logic circuit.

[A OR B] Bitwise OR

The ORing of two bits may be achieved through the use of an OR gate. For the case of operands of two bit widths (A and B), ORing is achieved through ORing of bits 1 of A and B (A0,B0), forming output bit 1 (Q0) followed by bits (A1,B1), forming output bit 2 (Q1).
Figure 13. depicts the final circuit implementing bitwise OR for operand A and B.

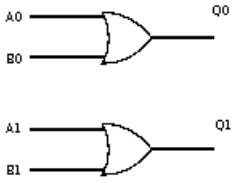

Fig.13 OR AND logic circuit.

8-to-1 multiplexer

With the basic logic blocks complete, it is necessary to implement some logic which may selectively output the result of the arithmetic operation currently defined by the opcode register.
For example, if the opcode defined by F=000 exists we would like to output the result of the arithmetic (A+B) operation to the output only.
This operational isolation may be achieved using a multiplexer. An 8-to-1 multiplexer may be constructed with 2 input multiplexers as indicated in Fig. 14.

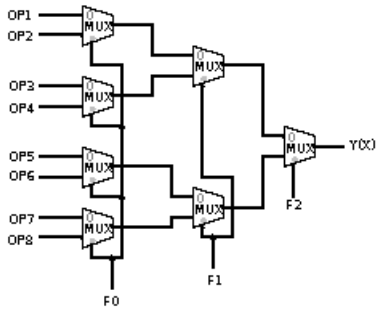

Fig.14 8-to-1 multiplexer.

Where the inputs OP1,OP2,OP(n)... represent a single output bit of each sub-circuit mentioned above, therefore two 8-to-1 multiplexers are required to represent both output bits Y0 and Y1. Each output bit is selected using three control bits (F0,F1,F2), these are the operational code register bits of our ALU

ALU

The above mentioned sub-systems are now connected to form the complete simplified ALU as depicted in Fig.15. The ALU consists of two input registers A and B whose individual bits may be set via (B1,B0,A1,A0). The opcode register is seen consisting of three bits (F0,F1,F2).
The operational mode of the ALU is set by these bits and their respective arithmetic operations are defined in Fig.2.
Output bits are represented by Y1 and Y0.

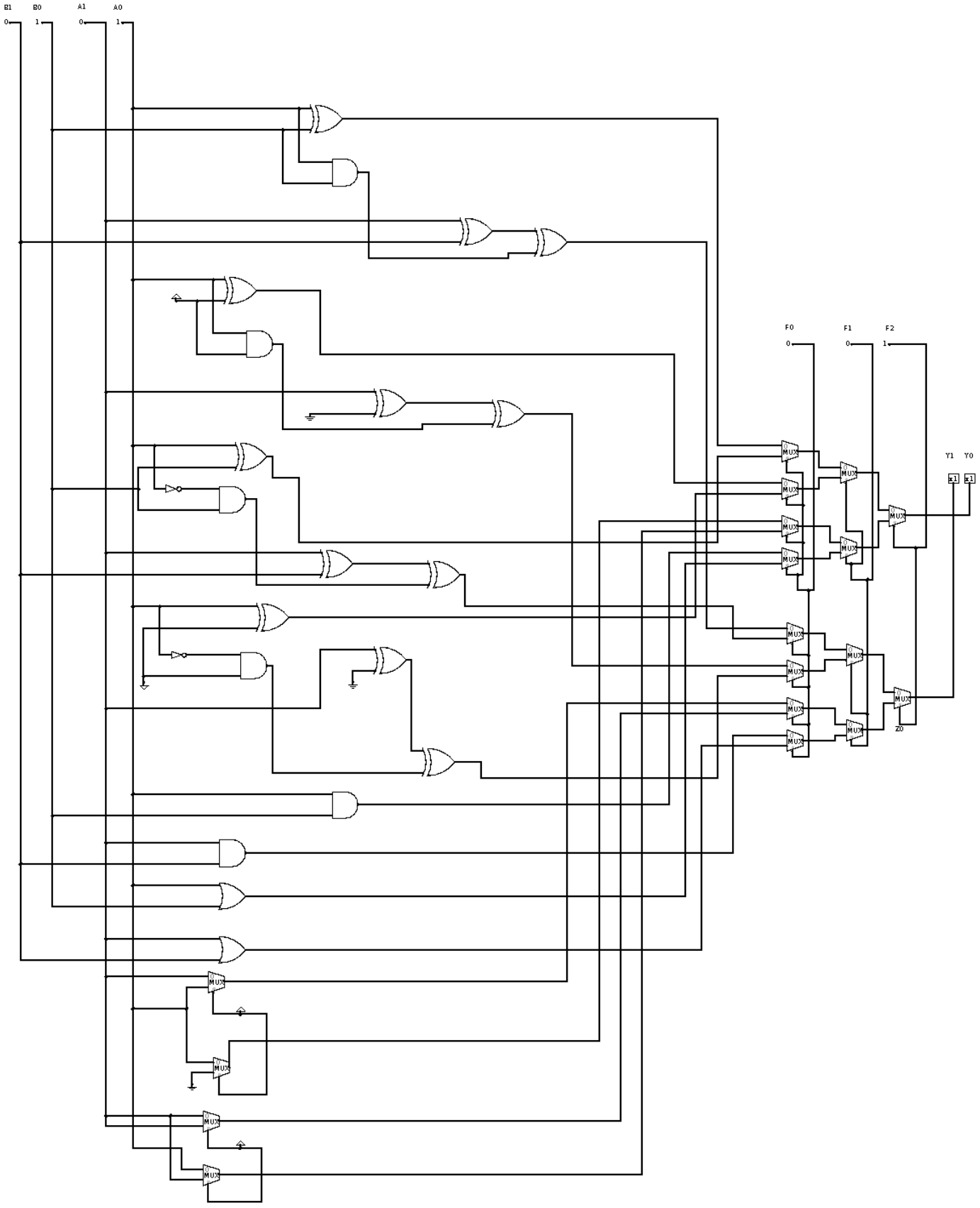

Fig 15. Complete arithmetic logic unit.