02 산술 논리 연산 장치

❖ **산술 논리 연산 장치**(Arithmetic Logic Unit, ALU) : 산술 연산과 논리 연산

- 주로 정수 연산을 처리
- 부동 소수(Floating-point Number) 연산 : FPU(Floating-Point Unit)
- 최근에는 ALU가 부동 소수 연산까지 처리

❖ 산술 연산 : 덧셈, 뺄셈, 곱셈, 나눗셈, 증가, 감소, 보수

❖ 논리 연산 : AND, OR, NOT, XOR, 시프트(shift)

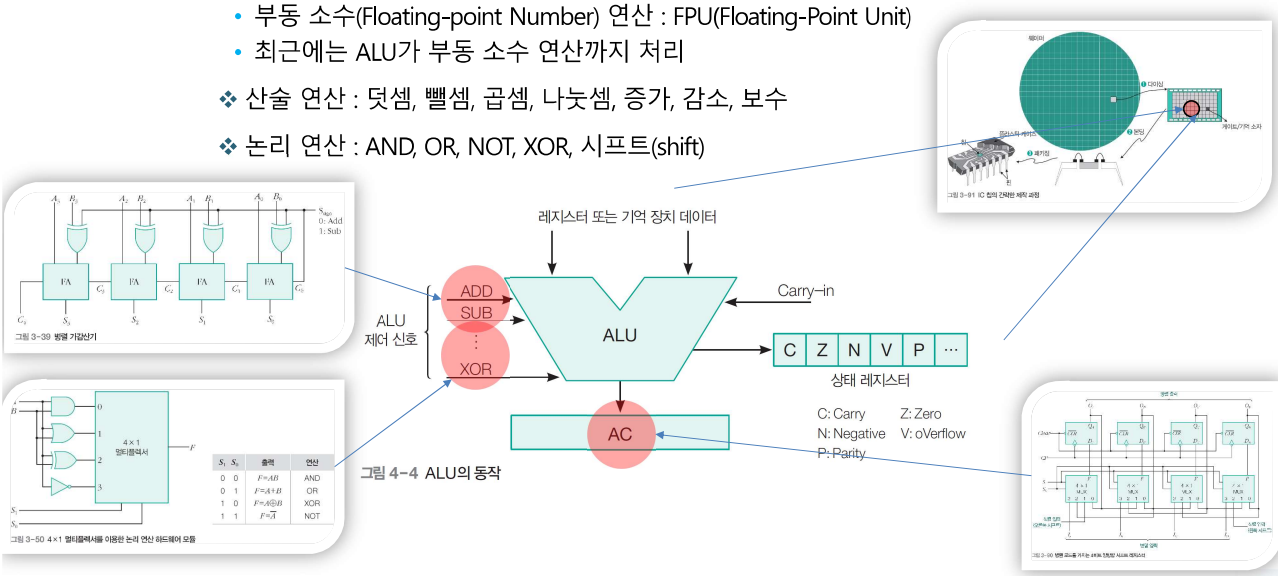


그림 4-4 ALU의 동작

02 산술 논리 연산 장치

1 산술 연산

표 4-1 산술 연산

연산	8비트 연산	
	동작	설명
ADD	X ← A + B	A와 B를 더한다.
SUB	X ← A + (~B + 1)	A + (B의 2의 보수)
MUL	X ← A * B	A와 B를 곱한다.
DIV	X ← A / B	A와 B를 나눈다.
INC	X ← A + 1	A를 1 증가시킨다.
DEC	X ← A − 1(0xFF)	A를 1 감소시킨다.
NEG	X ← ~A + 1	A의 2의 보수다.

곱셈을 빠르게 하는 방법이 필요함 → Booth 알고리즘

- 정수의 곱셈 → 이진수로 변환하여 곱셈
- 피승수(Multiplicand; M), 승수(Multiplier; Q)
- 예) 22 x 17

```
   22
 x 17
------
  154
   22
------
  374
```

```
      010110 (22)
×     010001 (17)
-------------------
      010110
+ 010110
-------------------
  0101110110
```

이진수 곱셈:
승수의 1이 있는 경우 피승수 전체를 위치시키고, 이를 모두 더함

22 x 14

```
      010110 (22)
×     001110 (14)
-------------------
      010110
    010110
  010110
-------------------
  0100110100
```

이진수 곱셈:
승수에 1이 많으면 덧셈을 많이 해서 느려지는 문제

- 1이 연속적으로 나오는 경우가 많다면
  그만큼 덧셈연산을 수행해야 하기때문에 곱셈연산이 느려지는 문제를 개선한 알고리즘 필요
- 1950년 영국 Andrew Donald Booth에 의해 개발 → 덧셈 횟수를 줄이는 알고리즘

Booth 알고리즘

https://www.grahn.us/projects/booths-algorithm/

```
   22
 x 17
------
  154
   22
------
  374
```

```
      010110 (22)
×     010001 (17)
-------------------
      010110
    010110
-------------------
  0101110110
```

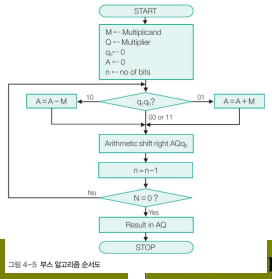


그림 4-5 부스 알고리즘 순서도

22 x 14

```
      010110 (22)
×     001110 (14)
-------------------
      010110
    010110
  010110
-------------------
  0100110100
```

308

Binary Multiplication Using Booth's Algorithm.

Enter any two integer numbers into the form and click 'Multiply' to watch Booth's algorithm run its magic.

Input

22 x 17 Multiply

Result

A	Q	Q_{-1}	M	Log
000000	010001	0	010110	Populate Data
101010	010001	0	010110	A = A - M
110101	001000	1	010110	Shift
001011	001000	1	010110	A = A + M
000101	100100	0	010110	Shift
000010	110010	0	010110	Shift
000001	011001	0	010110	Shift
101011	011001	0	010110	A = A - M
110101	101100	1	010110	Shift
001011	101100	1	010110	A = A + M
000101	110110	0	010110	Shift

plication Using Booth's Algorithm.

Enter any two integer numbers into the form and click 'Multiply' to watch Booth's algorithm run its magic.

Input

22 x 14 Multiply

Result

A	Q	Q_{-1}	M	Log
000000	001110	0	010110	Populate Data
000000	000111	0	010110	Shift
101010	000111	0	010110	A = A - M
110101	000011	1	010110	Shift
111010	100001	1	010110	Shift
111101	010000	1	010110	Shift
010011	010000	1	010110	A = A + M
001001	101000	0	010110	Shift
000100	110100	0	010110	Shift

02 산술 논리 연산 장치

❖ **Booth Algorithm**

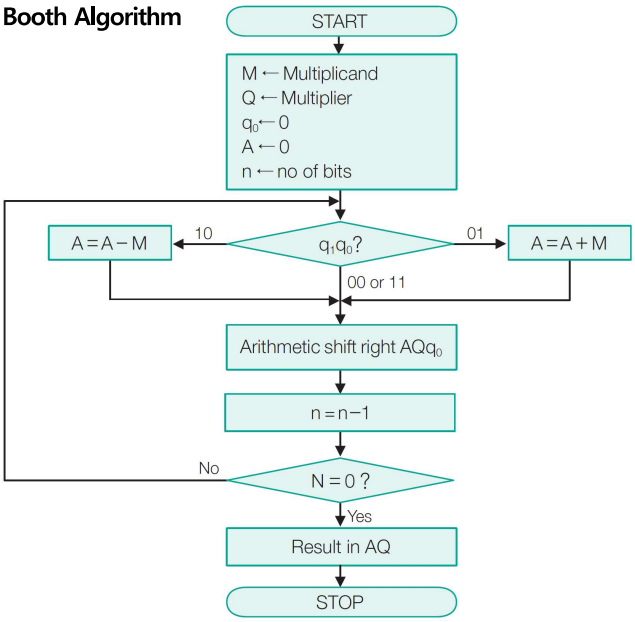
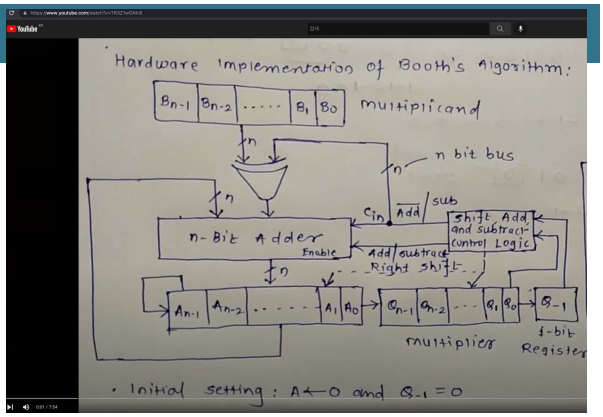
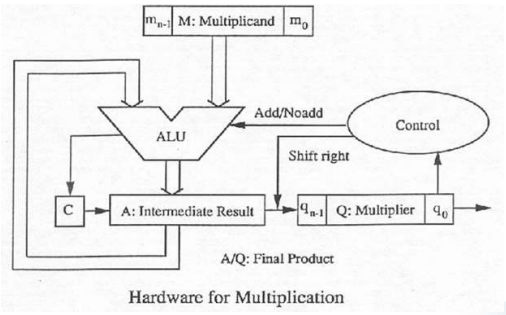


그림 4-5 부스 알고리즘 순서도





Hardware for Multiplication

02 산술 논리 연산 장치

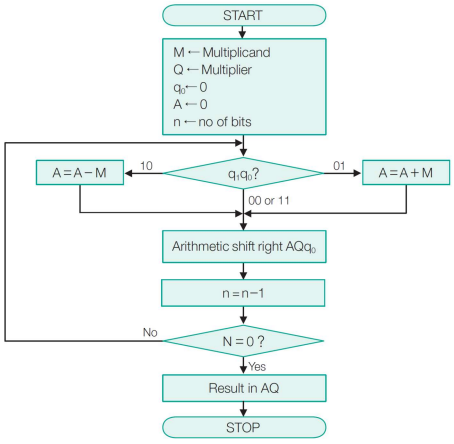


그림 4-5 부스 알고리즘 순서도

❖ **Booth Algorithm 예** : 5 * (-4)

n	A	$Q(q_4q_3q_2q_1)$	q_0	설명
4	0000	1100	0	초기 상태
3	0000	0110	0	q_1q_0이 00이므로 연산 없이 AQq_0을 오른쪽 산술 시프트
2	0000	0011	0	q_1q_0이 00이므로 연산 없이 AQq_0을 A오른쪽 산술 시프트
1	1011	0011	0	q_1q_0이 10이므로 A=A−M=0000+1011
	1101	1001	1	AQq_0을 오른쪽 산술 시프트
0	1110	1100	1	q_1q_0이 11이므로 연산 없이 AQq_0을 오른쪽 산술 시프트

Booth 알고리즘

```
   22
x  17
-----
  154
   22
-----
  374
```

```
      010110 (22)
x     010001 (17)
-----------------
      010110
     010110
-----------------
0101110110
```

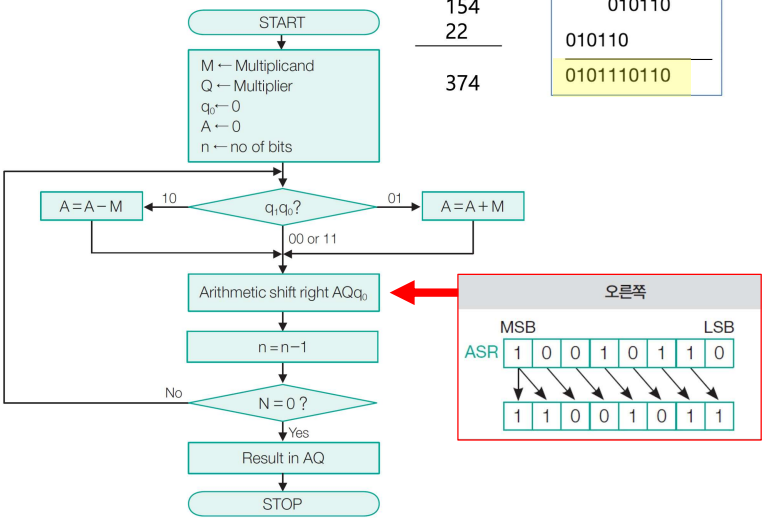
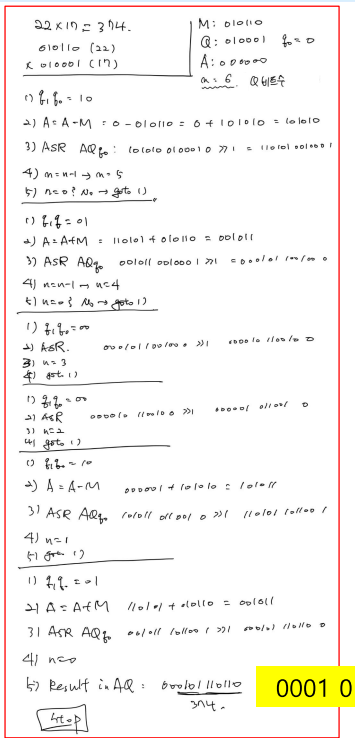


그림 4-5 부스 알고리즘 순서도



02 산술 논리 연산 장치

❖ **Booth Algorithm 예** : (-7) * (+3) = **-21 (-0001 0101 -> 2의보수 1110 1011)**;

M = 1001(-7), Q 0011(3); n=4

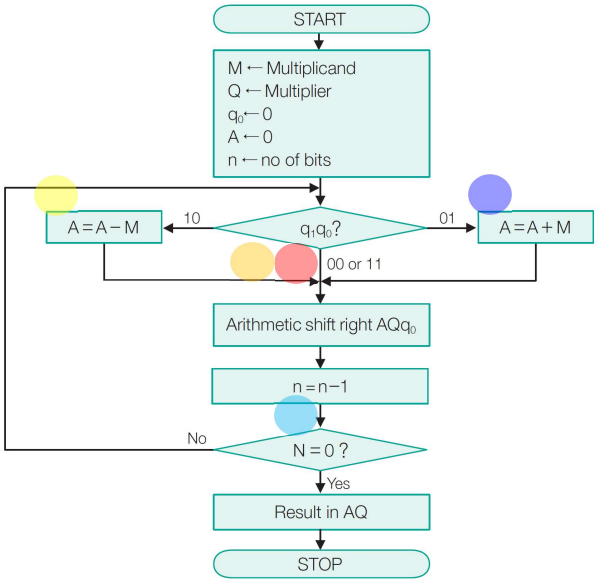


그림 4-5 부스 알고리즘 순서도

2 논리 연산과 산술 시프트 연산

표 4-2 논리 연산

연산	8비트 연산	
	동작	설명
AND	X←A&B	A와 B를 비트 단위로 AND 연산한다.
OR	X←A|B	A와 B를 비트 단위로 OR 연산한다.
NOT	X←~A	A의 1의 보수를 만든다.
XOR	X←A^B	A와 B를 비트 단위로 XOR 연산한다.
ASL	X←A≪n	왼쪽으로 n비트 시프트(LSL과 같다.)
ASR	X←A≫n, A[7]←A[7]	오른쪽으로 n비트 시프트(부호 비트는 그대로 유지한다.)
LSL	X←A≪n	왼쪽으로 n비트 시프트
LSR	X←A≫n	오른쪽으로 n비트 시프트
ROL	X←A≪1, A[0]←A[7]	왼쪽으로 1비트 회전 시프트, MSB는 LSB로 시프트
ROR	X←A≫1, A[7]←A[0]	오른쪽으로 1비트 회전 시프트, LSB는 MSB로 시프트
ROLC	X←A≪1, C←A[7], A[0]←C	캐리도 함께 왼쪽으로 1비트 회전 시프트
RORC	X←A≫1, C←A[0], A[7]←C	캐리도 함께 오른쪽으로 1비트 회전 시프트

❖ **논리 연산 예 1** : A=46=$00101110_{(2)}$, B=-75=$10110101_{(2)}$

A AND B		A OR B		A XOR B	
00101110	46	00101110	46	00101110	46
& 10110101	−75	| 10110101	−75	^ 11111111	−128
00100100	36	10111111	−65	11010001	−47

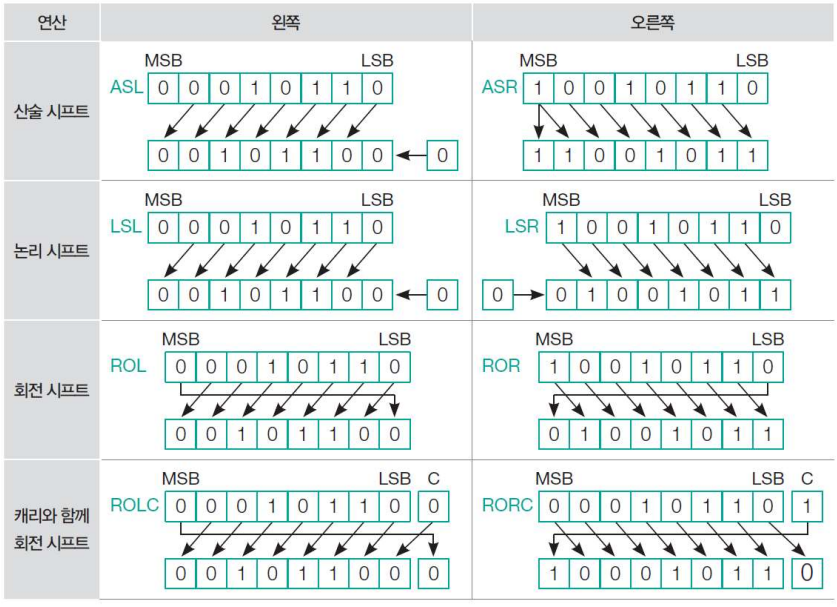
02 산술 논리 연산 장치

❖ 논리 연산 예 2

A AND B		A OR B	
00101110		00001110	
& 00001111	상위 4비트 삭제	| 10110000	상위 4비트 값 설정
00001110		10111110	

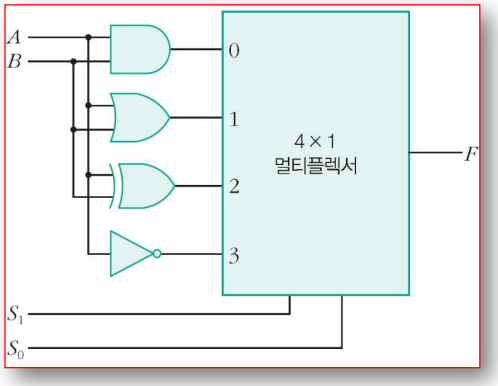
02 산술 논리 연산 장치

❖ 시프트 연산 예

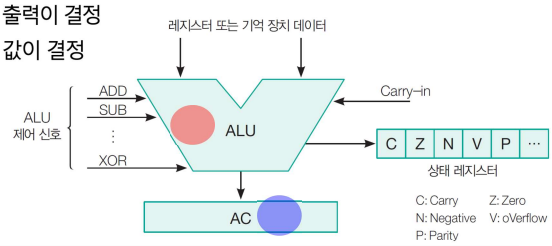


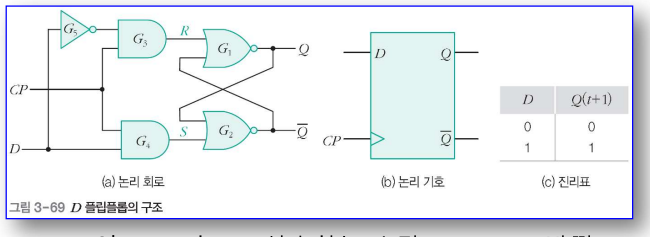
조합논리회로 vs. 순서논리회로(조합논리+기억)

- 조합 논리 회로(combinational logic circuit) : 이전 입력 값에 관계없이 현재 입력 값에 따라 출력이 결정
- 순서 논리 회로(sequential logic circuit) : 현재의 입력 값과 이전 출력 상태에 따라 출력 값이 결정



A,B,S_1,S_0 신호가 바뀌는 순간 F도 바로 바뀜





D=1이고 CP가 1로 상승하는 순간 Q(t+1)=1 바뀜
CP가 0인 동안은 Q(t+1) 상태 유지(기억)

CPU(ALU)는 빠르고 메모리는 느리기 때문에 계산 결과를 유지(기억)할 소자가 필요 → 레지스터