3 장

Half/Full Adder
반 가산기와 전 가산기
& 4-비트 입력 비교기

1. 실험 목표

▣ Half Adder 와 Full Adder 의 구성과 동작 원리를 이해한다.

▣ Adder 를 이용하여 간단한 논리회로를 직접 구성해 본다.

▣ AOI 구조를 이용하여 4-비트, 2 입력 비교기를 설계한다.

2. 실험 이론

논리회로의 가장 기본적인 연산기능은 이진수의 합을 구하는 Adder 이다. 디지털 회로는 여러 비트를 하나의 처리 단위로 묶어 한 번에 처리하며, 8-비트, 16-비트, 32-비트, 64-비트 등이 가능하다. 덧셈 연산을 수행하는 경우 단위 유닛 n-비트는 1-비트 Adder 를 n 개 사용하여

구성하게 된다. 이 경우 각 비트 위치에서 보면 비트-0 위치에는 2 개의 입력이 존재하나, 그 이후 비트 위치에서는 아래 단에서 올라오는 올림수 (carry) 입력이 추가로 필요하여, 3 개의 입력이 필요하게 된다. 따라서, 단순히 2bit 의 합을 구하는 회로는 Half Adder 라 정의하고, 올림수 처리가 가능한 3 개의 입력이 필요한 Adder 는 Full Adder 라고 정의한다.

가) Half Adder 설계

Half Adder 는 이진수의 두 입력 X 와 Y 에 대한 합 (S)와 자리올림의 값 (Cout)을 출력 값으로 구하는 회로로서, 낮은 자리에서 올라오는 올림수는 고려하지 않고, 단지 두 개의 이진수만을 가산한다. [표 3-1]은 Half Adder 의 진리표이며, 모든 가능한 입력의 경우의 수에 대하여 정의된 결과값을 구하여 진리표를 구하면 된다.

표 3-1. Half Adder 의 진리표.

입력		출력	
X	**Y**	**Cout(올림수)**	**S(합)**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

이진수의 덧셈을 수행하면 위의 표와 같이 진리표를 얻게되며, 합을 S, 자리 올림수를 C 라 하면, 진리표로부터 각 출력에 대한 논리식을 아래와 같이 유도할 수 있다. 예를 들면, 합 S 는 출력이 참인 경우에 대해 해당 입력 변수 조합에 대해 기술하고, 각 각의 참인 경우에 대해 OR 로 변수 조합을 표시하면 된다.

S = (X and Y') or (X' and Y) = X xor Y
Cout = X and Y

위에 주어진 논리식으로부터 논리회로를 구성하면 [그림 3-1]과 같다.

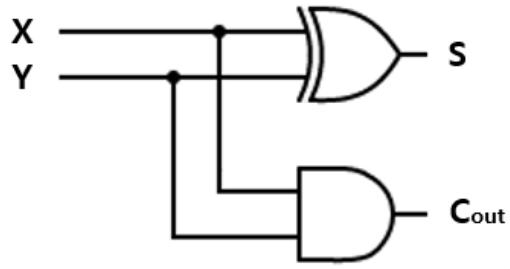

그림 3-1. Half Adder 의 구성.

나) Full Adder 설계

임의의 비트로 구성된 수를 더하기 위하여, 여러 비트의 덧셈 수행이 필요한 경우에는 두 입력 수 외에 낮은 자리에서 올라오는 올림수도 가산해야 하므로 Half Adder 만으로는 연산기능을 구현하기에 부족하다. Full Adder 는 두 개의 이진수와 낮은 자리에서 발생한 올림수 (Cin)까지 고려하여 입력 세 개의 합과 올림수 (Cout)을 구하는 논리회로이다. Full Adder 의 진리표는 [표 3-2]의 진리표와 같다.

표 3-2. Full Adder 의 진리표.

입력			**출력**	
Cin	X	**Y**	**Cout**	**S(합)**
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

위 [표 3-2]의 진리표에 의해 출력의 값이 참인 경우에 대해 논리식을 유도하면,

S = (X and Y′ and Cin) or (X′ and Y and Cin′) or (X′ and Y′ and Cin) or (X and Y and Cin)
= X xor Y xor Cin

Cout = (X and Y) or (X and Cin) or (Y and Cin)

논리식으로부터 해당 논리회로를 구성하면 [그림 3-2]와 같다.

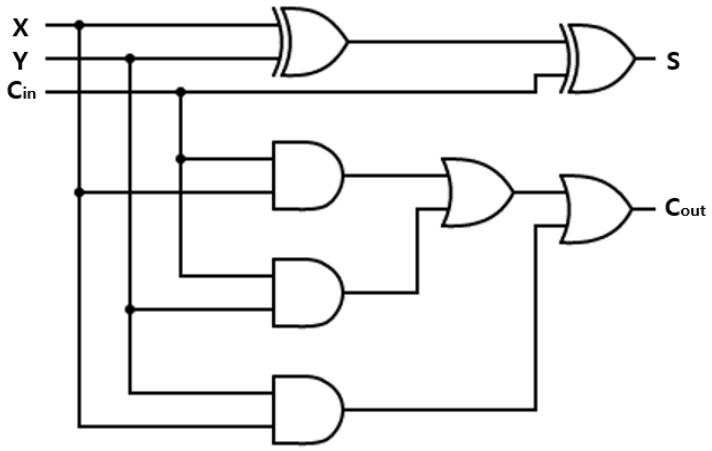

그림 3-2. Full Adder 구성.

다) 4-비트 입력 비교기 설계

2 개의 입력 값이 같은 지, 다른 지를 간단하게 비교하여 출력을 제시하는 회로를 설계하자. 이 실험에서는 2 개의 입력은 각각 A (A3A2A1A0)와 B (B3B2B1B0)로 표시하고, 각각은 4-비트로 구성된 수를 가정한다. 2 개의 입력 값이 동일하기 위하여서는 각각의 비트 위치의 값이 동일하여야 한다. AOI 패키지를 이용하기 위하여 F′을 구하여 출력 F 를 구하게 된다. 예를 들어 F = A xor B = A′ B + A B′를 구하기 위하여, F′ = XNOR = A′B′ + AB 을 구하여 다음 [그림 3-3]과 같이 AOI 패키지에 연결한다.

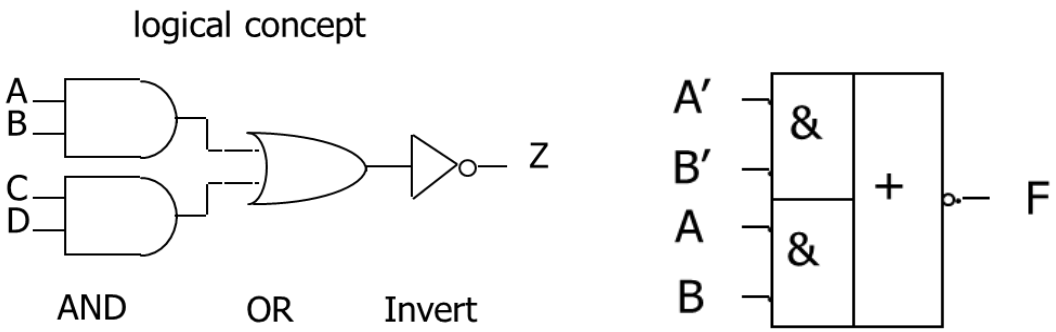

그림 3-3. 2x2 AOI 기반의 2-비트 비교기 구성.

4-비트 비교기를 구하기 위하여 주어진 4-비트 각각의 비트 값에 대한 비교를 수행하고 모든 비트 위치에서 입력 값이 동일하면 같다고 할 수 있다. 따라서 출력, Z = (A0 B0 + A0′ B0′)(A1 B1 + A1′ B1′)(A2 B2 + A2′ B2′)(A3 B3 + A3′ B3′)로 구할 수 있으며, 각각의 비트 위치에 2x2 AOI 모듈을 사용하여 다음 [그림 3-4]와 같은 회로를 구성할 수 있다.

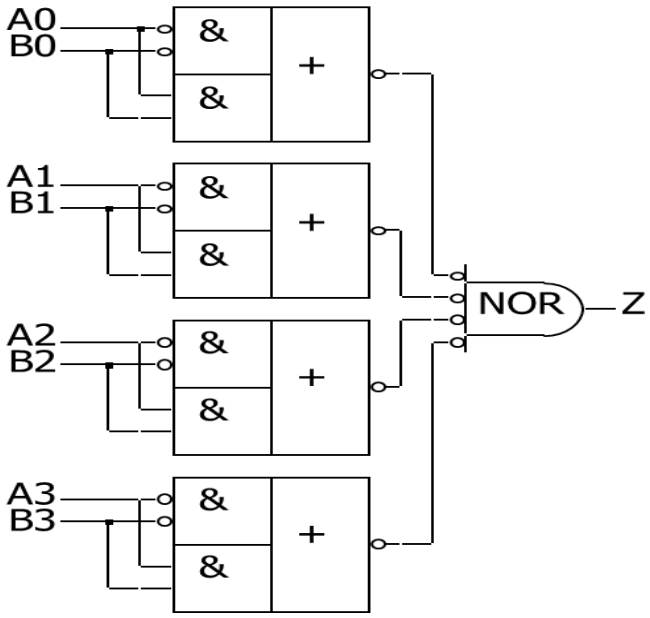

그림 3-4. 2x2 AOI 를 이용한 4-비트 입력 비교기 구성.

3. 예비보고서 작성

▣ 카르노 맵을 이용하여 진리표(Half Adder / Full Adder)를 논리식으로 만드시오.

▣ Half Adder 를 이용하여 Full Adder 를 구성해 보시오.

▣ 2 x 2 AOI 구조를 이용하여 4-비트 2 입력 비교기를 구성해 보시오.

▣ Verilog HDL 문법을 학습하시오.

4. 실험에 필요한 기기

▣ 실험용 PC

▣ MAX PLUS II 소프트웨어 / Quartus II-Modelsim 소프트웨어

5. 실험 내용과 과정

① 예비보고서에서 작성한 카르노 맵을 통해 얻은 논리식으로 Half/Full Adder 를 Verilog HDL 로 설계한다.
② 마찬가지로, 2-input / 4-bit 입력 비교기의 설계 도면을 바탕으로, Verilog HDL 코드를 작성한다.
③ Modelsim-Altera 를 구동하여, 위에서 작성한 Verilog HDL 코드의 컴파일을 수행한다.
④ 컴파일을 수행한 후 에러가 발생하였다면 이를 해결하고, 문제가 없다면 Testbench 코드 작성을 통해 Simulation 을 수행한다.
⑤ 결과 화면을 확인한 후, [6. 실험 결과 보고서]에서 제시한 질문들에 대해서 고찰해본다

6. 실험 결과 보고서

① 컴파일 수행 시에 에러가 발생하였다면, 그 원인에 대해서 분석하시오.

② 글리치가 발생한 원인에 대해서 논하고, 그 해결책을 찾아보시오.

③ 설계한 회로의 최대 경로지연을 찾아서, 클럭에 의해서 동작시킬 경우의 최대 동작 주파수를 구해보시오.

④ Full Adder 의 구현 물을 바탕으로, 4-bit + 4-bit 가산기를 구현해본 후 이를 검증해본다. (Full Adder Module 의 재활용을 통해 4-bit 가산기를 구현하시오.)

⑤ (4)에서 설계한 4-bit 가산기의 최대 지연 시간은 어느 정도 인가? 그리고 (3)에서의 결과와 연관 지었을 때, n-bit 가산기에 대한 지연 시간이 어느 정도일지 일반화하여 본다.

⑥ 2-input / 4-bit Comparator(비교기)의 사용에 있어서, 왜 일반적인 AND, OR, NOT gate 를 사용하지 않고, AOI 라는 특수한 gate 를 사용하는가? 그 이유를 타당하게 설명해보시오. (힌트: 로직 설계에 있어서 MOSFET Transistor 의 사용 개수는 작을수록 좋다.)