컴퓨터 공학 기초 설계 및 실험1

예비 & 결과 보고서

실험제목: Half-adder & full-adder Multiplexer and de-multiplexer

실험일자: 2018년 05월 24일 (금)

제출일자: 2018년 05월 30일 (목)

학 과:컴퓨터정보공학부

담당교수: 이혁준

실습분반: 03-금012

학 번: 2018202074

성 명: 김상우

결과보고서

1. 제목 및 목적
   1. 제목

Half-adder & full-adder Multiplexer and de-multiplexer

* 1. 목적

반가산기(Half-adder)와 전가산기(full-adder)의 회로를 직접 구현하고 이를 실행시켜봄으로서 그것들의 개념과 차이를 이해한다. (추가적으로 전감산기도 구현한다.) 마찬가지로 멀티플렉서(Multiplexer)와 디멀티플렉서(de-multiplexer)를 구현하고 그들의 개념을 이해하며 둘의 차이를 이해한다. 결과적으로 구현해본 경험을 바탕으로 이들을 이용해서 다른 회로를 구현하는 응용력을 키우고 이전까지 만들 수 있었던 회로보다 더욱 효율적인 회로를 만들 수 있게 되는 것이 최종적인 목표이다. 특히 이전까지 배운 것과 연계되므로 이전까지 배운 지식들에 대한 복습도 목적으로서 겸한다.

1. 실험 결과

-(실험 6.2) 반가산기 구현 및 입력에 따른 출력 확인



-논리식

S = A (+) B = ( A’ B )+ A B’ (\*XOR을 (+)로 표현한다.)

C = ( ( A \* B )’ )’ = A \* B

|  |  |  |  |
| --- | --- | --- | --- |
| 입력 | | 출력 | |
| A | B | S | C |
| 0 | 0 | 0 | 0 |
| 0 | +5 | 1 | 0 |
| +5 | 0 | 1 | 0 |
| +5 | +5 | 0 | 1 |

정리: S는 XOR회로로 A와 B값이 같을 때 0, 다를 때 1을 출력한다.

C는 A와 B를 NAND 후 NOT을 붙여준 값으로 AND를 해준 것의 값과 같다.

-(실험 7.1) 전가산기 구현 및 출력전압 측정



-논리식

Sn = ( An (+) Bn ) (+) Cn-1 = An (+) Bn (+) (Cn-1)

Cn = (( ( An (+) Bn ) \* (Cn-1) )’ \* ( A \* B )’ ) = ( An\*Bn ) + ( ( A(+)B )\*Cn-1 )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 입력 | | | 출력 | |
| An | Bn | Cn-1 | Sn | Cn |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | +5 | 1 | 0 |
| 0 | +5 | 0 | 1 | 0 |
| 0 | +5 | +5 | 0 | 1 |
| +5 | 0 | 0 | 1 | 0 |
| +5 | 0 | +5 | 0 | 1 |
| +5 | +5 | 0 | 0 | 1 |
| +5 | +5 | +5 | 1 | 1 |

정리:Sn은 3-input XOR과 같다.( 0이 홀수일 경우 0, 0이 짝수일 경우 1 )

Cn은 위 같은 논리식을 가지며, An과 Bn이 +5V거나 Cn-1이 +5V이고 An과 Bn 중 하나라도 +5V면 1을 가진다.

-(실험 7.3) 전감산기 구현 및 출력전압 측정

논리식: Bn+1= ( An (+) Bn )’ \* bn + An’ \* Bn

d = ( An (+) Bn ) (+) bn = An (+) Bn (+) bn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 입력 | | | 출력 | |
| An | Bn | bn | Bn+1 | D |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | +5 | 1 | 1 |
| 0 | +5 | 0 | 1 | 1 |
| 0 | +5 | +5 | 1 | 0 |
| +5 | 0 | 0 | 0 | 1 |
| +5 | 0 | +5 | 0 | 0 |
| +5 | +5 | 0 | 0 | 0 |
| +5 | +5 | +5 | 1 | 1 |

정리: Bn+1은 An이 0일 때 1, An이 1일 때 0을 출력한다. 단, 모든 입력이 0일땐 0, 모든 입력이 +5일땐 1을 출력한다.

D의 경우 0의 개수가 홀수 일시 0, 짝수일 경우 1을 출력한다.

-(실험 16.1) 7420, 7404 게이트 기반 4채널 멀티플렉서 회로의 입력에 따른 출력값

논리식: Y=E’\*(S1’\*S2’\*I0\*I1’\*I2’\*I3’+S1’\*S2\*I0’\*I1\*I2’\*I3’+

S1\*S2’\*I0’\*I1’\*I2\*I3’+ S1\*S2\*I0’\*I1’\*I2’\*I3)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 입력 | | | | | | | 출력 |
| E | S1 | S2 | I0 | I1 | I2 | I3 | Y |
| +5 | X | X | X | X | X | X | 0 |
| 0 | 0 | 0 | 0 | +5 | +5 | +5 | 0 |
| 0 | 0 | 0 | +5 | 0 | 0 | 0 | 1 |
| 0 | 0 | +5 | 0 | +5 | 0 | 0 | 1 |
| 0 | 0 | +5 | +5 | 0 | +5 | +5 | 0 |
| 0 | +5 | 0 | 0 | 0 | +5 | 0 | 1 |
| 0 | +5 | 0 | +5 | +5 | 0 | +5 | 0 |
| 0 | +5 | +5 | 0 | 0 | 0 | +5 | 1 |
| 0 | +5 | +5 | +5 | +5 | +5 | 0 | 0 |

정리: S1을 2진수의 2의 자리 S2를 1의 자리로 가정하자. 이 값을 십진수로 나타낼 때의 값을 n이라 할 때, In이 1이고 나머지 I들이 0일경우 1을 출력한다.

이는 E가 0일 때를 가정하며 E가 +5라면 무조건 0을 출력한다.

-(실험 16.3) 7411, 7404 게이트 기반 4채널 디멀티플렉서의 입력에 따른 출력값



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 입력 | | | 출력 | | | |
| E | S1 | S2 | I0 | I1 | I2 | I3 |
| 0 | X | X | 0 | 0 | 0 | 0 |
| +5 | 0 | 0 | 1 | 0 | 0 | 0 |
| +5 | 0 | +5 | 0 | 1 | 0 | 0 |
| +5 | +5 | 0 | 0 | 0 | 1 | 0 |
| +5 | +5 | +5 | 0 | 0 | 0 | 1 |

정리: S1을 2진수의 2의 자리 S2를 1의 자리로 가정하자. 이 값을 십진수로 나타낼 때의 값을 n이라 할 때, In이 1이고 나머지 I들이 0일경우 1을 출력한다.

이는 E가 +5일 때를 가정하며 E가 0라면 무조건 0을 출력한다.

1. 고찰

위 실험들에서 사용된 가(감)산기들은 수를 세는 데에 있어 상당히 유용하게 사용될 수 있다. 그렇기에 앞으로 회로를 짬에 있어 이를 유용하게 사용할 수 있을 것입니다. 특히 연속적인지 비연속적인지에 따라 전가산기, 반가산기를 적절하게 사용하는 것도 중요할 것입니다.

실험 16-1, 16-3에서 사용된 회로들은 지금까지의 입력에 의한 결과를 출력하기 위한 회로보다는 선택에 최적화된 회로였습니다. 이는 컴퓨터 언어에서 Switch문과 유사합니다. 또한 2진수와 유사하게 2^(S의 개수) 만큼의 선택지를 만들 수 있습니다.(n채널) 이를 이용하면 지금보다 다양하고 유동적인 회로를 짤 수 있을 것입니다. 다만 그만큼 입력값이 많아지므로 필요와 재료 등에 따라 조절할 수 있는 능력이 필요할 것입니다. 실제로 디멀티플렉서는 멀리플렉서보다 입력값의 수가 더 적은걸 알 수 있습니다.

위 회로들을 구현하면서 책과 실제 회로가 다른 경우를 발견하였습니다. (실험 7.3에 NOT GATE가 없다.) 이러한 점은 앞으로도 다시 일어날 가능성이 충분합니다.(회로가 무조건 알맞게 작동하지 않으므로) 앞으로는 이러한 점을 고려하여 회로를 짜기 전에 회로를 따라 결과값을 예상하여 회로가 맞는지를 확인하는 등의 노력이 필요할 것입니다.

예비보고서

1. 제목 및 목적
   1. 제목

Adder & subtractor using 2’s complement

* 1. 목적

보수의 개념과 보수와 관련된 공식, 사용 방안 등을 이해한다. 또한 배운 보수의 개념을 활용하여 가(감)산기를 구현하여 보고 이를 통해서 이전보다 더 다양한 형태의 가(감)산기를 구현할 수 있는 능력을 키우는 것을 목적으로 한다. 추가적으로 가(감)산기들은 다른 용도로 쓰일 수 있으므로 이를 통해 회로를 더 유동적으로 만들 수 있는 능력 또한 키우며 동시에 회로를 읽어내는 능력도 키우는 것 또한 또 하나의 목적이다.

1. 원리(배경지식)

**보수:**

뜻은 “각자리의 숫자의 합이 어느 일정한 수가 되게 하는 수”이다. 즉, n에 대한 m의 보수는 n-m을 의미한다. 예를 들어 10에 대한 3의 보수는 10-3인 7이 되는 것이다.

가장 유용하게 쓰이는 보수는 2의 보수이다.

**2의 보수:**

2의 보수는 1의 보수에 1을 더한 것을 의미하기도 한다. 즉 1의 보수는 2의 보수보다 1이 작음을 의미한다. 예를 들어 2진수 0101110의 보수는 1010001+1=1010010이 될 것이다.

**병렬 가감산기:**

가산과 감산의 연산을 하는 회로로, 2진 가산기라고도 한다.

가산과 감산의 연산은 하나의 2진 가산기로 된 하나의 회로로 결합될 수 있는데, 전가산기에 XOR게이트를 포함시킴으로서 이것이 구현된다.



위에서 M은 연산을 제어하게 된다.

M이 0이면 회로는 가산기의 형태를, M이 1이면 회로는 감산기의 형태를 띄게 된다.

위 회로에서는 Exclusive-OR 게이트(XOR 게이트)는 입력 M와 입력 B의 한 비트를 받아들이게 되는데, 예를 들어 M=0이면 B(+)0=B이므로 전가산기는 B의 값을 받게 된다. 이때 입력 캐리는 0이 된다. M=1일 때 B(+)0=B’이므로 C1은 1이 된다. 이에 따라 입력 B는 전부 보수가 되며 입력 캐리는 1이 되어 1을 더하게 된다.

최종적으로 회로는 A의 B에 대한 2의 보수를 더하는 연산을 행하게 된다.

**BCD 가산기:**

2개의 BCD 숫자를 4비트 2진식 가산기에 인가한다고 가정하자, 이 가산기는 2진식으로합을 형성할 것이며 이는 0부터 19까지의 범위를 이르는 결과를 만들게 될 것이다.

C= K + Z8\*Z4 + Z8\*Z2 (Z(n)은 십진수 n을 나타내는 2진수의 자리의 값을 의미한다.)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2진수의 합 | | | | | BCD의 합 | | | | | 10  진수 |
| K | Z8 | Z4 | Z2 | Z1 | C | S8 | S4 | S2 | S1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| : | : | : | : | : | : | : | : | : | : | : |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 8 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 9 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 10 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 11 |

위 표와 같이 표현되며 위 같은 경우 2진수는 1/0011일 때 BCD의 합은 1/1001일 때 최대이다.(컴퓨터보단 사람이 인식하기 편하다.)

추가적으로 위를 보면 C=1일 때 2진수의 값에 0110을 더한 값과 같아짐을 알 수 있다.

1. 참고문헌

서적:

이원석,정길수/논리회로 실험/생능출판/2010

웹:

보수

/ <https://terms.naver.com/entry.nhn?docId=3405109&cid=47324&categoryId=47324>

ADDER(가산기)

/ <https://punkpunkpunk.tistory.com/80>

BCD가산기, CLA가산기

/ <https://m.blog.naver.com/tjdowl123/221191025593>