6주차 결과보고서

전공 : 국제한국학과 학년 : 4학년 학번 : 20181202 이름 : 김수미

**1. 실험 목적**

- Adder와 Subtractor의 개념을 이해한다.

- Code Converter(8421(BCD) to 2421)의 개념을 이해한다.

- Verilog 언어와 논리식을 사용해 다양한 Adder 및 Subtractor를 구현해 본다.

- Verilog 언와와 논리식을 사용해 Code Converter를 구현해 본다.

- FPGA를 통해 Verilog로 구현된 회로의 동작을 확인한다. (강의자료에는 실험 목표로써 기재되어 있으나 코로나 비대면 수업으로 인해 FPGA 실습은 진행하지 않았다).

**2. Full Adder 및 Half Adder 의 simulation 결과 및 과정에 대해서 설명하시오.**

1) Truth Table

|  |  |
| --- | --- |
| Full Adder | Half Adder |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | A | B |  | S(Sum) |  | | 0 | 0 | 0 | 0 | 0 | | 1 | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 1 | 0 | | 0 | 0 | 1 | 1 | 0 | | 1 | 1 | 0 | 0 | 1 | | 1 | 0 | 1 | 0 | 1 | | 0 | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 1 | 1 | | |  |  |  |  | | --- | --- | --- | --- | | A | B | S(Sum) | C(Carry) | | 1 | 1 | 0 | 1 | | 1 | 0 | 1 | 0 | | 0 | 1 | 1 | 0 | | 0 | 0 | 0 | 0 | |

2) Schematic

|  |
| --- |
| Full Adder |
|  |
| Half Adder |
|  |

3) Source Code

|  |  |  |  |
| --- | --- | --- | --- |
| Full Adder Design Source | Full Adder Test Bench | Half Adder Design Source | Half Adder Test Bench |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 영수증이(가) 표시된 사진  자동 생성된 설명 | 텍스트이(가) 표시된 사진  자동 생성된 설명 | 텍스트이(가) 표시된 사진  자동 생성된 설명 |

4) Simulation

|  |
| --- |
| Full Adder |
|  |
| Half Adder |
|  |

5) 결과 및 과정

Full Adder 는 입력값 두개와 Cin 값 까지 총 세개의 input을 고려하고, Sum 값과 Cout 값을 출력하지만 Half Adder는 입력값 두개만 고려하며 Sum 값과 Cout 값을 출력한다는 점에서 차이가 있다.

Full Adder에서 Sum 값은 입력값 두개와 Cin 값 세개에 XOR을 취한 값으로, 입력값이 A, B 라고 할 때 A, B, Cin 중에서 1값이 홀수개이면 Sum 값이 1이 됨을 알 수 있다.  
Cout 값은 식으로 나타내면 Cin(A^B)+AB 와 같다(^=XOR). 즉 입력값이 모두 1이거나, 또는 입력값 중 둘 중 하나만 1이고 Cin 값이 1인 경우 Cout 값은 1이 됨을 알 수 있다.

Half Adder에서 Sum 값은 입력값 두개에 XOR을 취한 값으로, 두 값 중 하나만 1 인 경우 Sum 값은 1이 된다. Carry 값은 입력값 두개에 AND를 취한 값으로, 두 값이 모두 1 인 경우 Carry 값이 1이 된다.

**3. Full Subtracter 및 Half Subtracter 의 simulation 결과 및 과정에 대해서 설명하시오.**

1) Truth Table

|  |  |
| --- | --- |
| Full Subtracter | Half Subtracter |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | A | B | bn-1 | D | bn | | 1 | 1 | 1 | 1 | 1 | | 1 | 1 | 0 | 0 | 0 | | 1 | 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 0 | 1 | | 0 | 0 | 1 | 1 | 1 | | 0 | 1 | 0 | 1 | 1 | | 1 | 0 | 0 | 1 | 0 | | 0 | 0 | 0 | 0 | 0 | | |  |  |  |  | | --- | --- | --- | --- | | A | B | D(Difference) | b(Borrow) | | 1 | 1 | 0 | 0 | | 1 | 0 | 1 | 0 | | 0 | 1 | 1 | 1 | | 0 | 0 | 0 | 0 | |

2) Schematic

|  |
| --- |
| Full Subtracter |
|  |
| Half Subtracter |
|  |

3) Source Code

|  |  |  |  |
| --- | --- | --- | --- |
| Full Subtracter Design Source | Full Subtracter Test Bench | Half Subtracter Design Source | Half Subtracter Test Bench |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 영수증이(가) 표시된 사진  자동 생성된 설명 | 텍스트이(가) 표시된 사진  자동 생성된 설명 | 텍스트이(가) 표시된 사진  자동 생성된 설명 |

4) Simulation

|  |
| --- |
| Full Subtracter |
|  |
| Half Subtracter |
|  |

5) 결과 및 과정

Full Subtracter 는 입력값 두개와 bn-1 값 까지 총 세개의 input을 고려하고, Differentce 값과 bn 값을 출력하지만 Half Subtracter 는 입력값 두개만 고려하며 Difference 값과 b 값을 출력한다는 점에서 차이가 있다.

Full Subtracter 에서 Difference 값은 입력값 두개와 bn-1 값 세개에 XOR을 취한 값으로, 입력값이 A, B 라고 할 때 A, B, bn-1 중에서 1값이 홀수개이면 Difference 값이 1이 됨을 알 수 있다.  
bn 값은 식으로 나타내면 (A^B)’bn-1 + A’B 와 같다(^=XOR). 즉 두 입력값이 0 또는 1로 동일하면서 bn-1값이 1일 때, 또는 A값이 0이고 B값이 1일 때 bn의 값은 1이 된다.

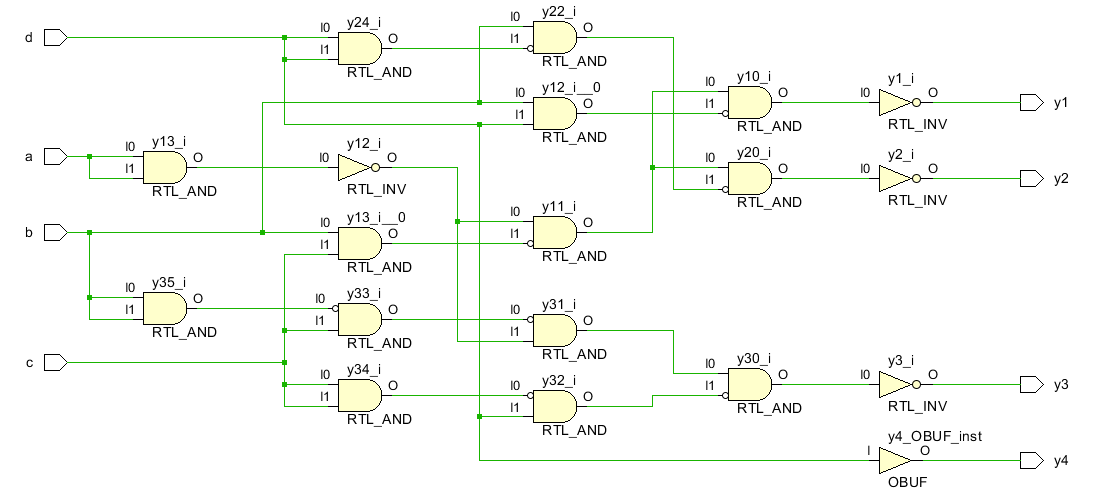
Half Subtracter의 경우 Dffierence는 두 입력값에 XOR를 취한 값이다. 즉 두 입력값이 서로 다른 경우 Difference 값은 1이 된다. Borrow 값은 두 입력값이 A와 B일 때 식으로 나타내면 A’B와 같다. 즉 A값이 0이고 B값이 1인 경우 Borrow값은 1이 된다.

**4. 8421(BCD)-2421 Code converter simulation 결과 및 과정에 대해서 설명하시오.**

1) Truth Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8421 (BCD) CODE | | | | 2421 CODE | | | | |
| A1 | B1 | C1 | D1 | A2 | B2 | C2 | D2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

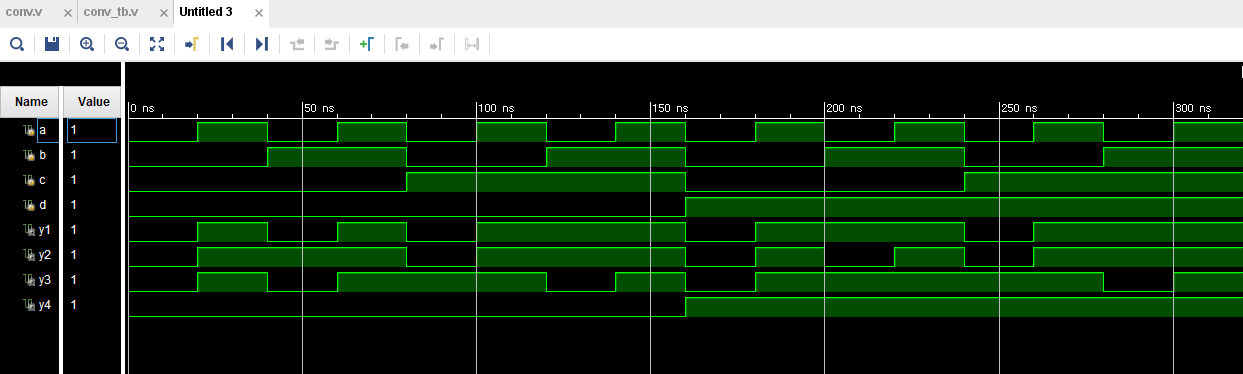
2) Schematic



3) Source Code

|  |  |
| --- | --- |
| Design Source | Test Bench |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 영수증이(가) 표시된 사진  자동 생성된 설명 |

4) Simulation



5) Karnaugh Map

|  |  |
| --- | --- |
| Karnaugh Map | |
| A2 | B2 |
|  |  |
| C2 | D2 |
|  |  |

|  |  |
| --- | --- |
| SOP Form | |
| A2 | B1D1 + B1C1 + A1C1’ |
| B2 | A1C1’ + B1D1’ + B1C1 |
| C2 | A1C1’ + B1’C1 + B1C1’D1 |
| D2 | D |
| POS Form | |
| A2 | (A1’+C1’+D1’)(A1’+B1’) |
| B2 | (A1’+C1’+D1)(A1’+B1’) |
| C2 | (A1’+C1’+D1’)(A1’+B1’+C1’)(B1+C1) |
| D2 | (C1’+D1’)(C1+D1’) |

6) 결과 및 과정

정의된 논리식이 없어 Truth Table을 바탕으로 카르노맵을 그려 직접 논리식을 도출해 내야 한다. 8421 BCD Code의 10가지 종류의 값의 조합을 바탕으로 2421 Code의 네가지 값에 대한 카르노맵을 그려 각각 논리식을 작성해 주면 된다.

**5. 결과 검토 및 논의사항**

- NAND, NOR 게이트만 활용하여 회로를 구성하면 비용을 훨씬 절감할 수 있지만, 논리식을 변형하는데 시간이 많이 소요되어 시도해 보지 않았다. NAND, NOR 게이트만을 이용하여 논리식을 짜보는 연습도 필요할 것 같다.

**6. 추가 이론 조사 및 작성**

1) NAND, NOR 게이트의 활용

아래 한가지 규칙만 기억하면 회로를 NAND, NOR 게이트로 변환하기 쉽다.

드모르간 법칙에 의해 (A’ + B’) = (A B)’ 이고, (A’ B’) = (A + B)’이다.

따라서 아래와 같은 OR 게이트는 NAND 게이트로 표현할 수 있고, AND 게이트는 NOR 게이트로 표현할 수 있다.

|  |
| --- |
| NAND, NOR 게이트의 활용 |
|  |