6주차 결과보고서

전공 : 컴퓨터공학과 학년 : 3학년 학번 : 20211558 이름 : 윤준서

**1. 실험 목적**

• Adder(가산기)와 Subtractor(감산기)의 개념을 이해한다.

• Code Converter(부호 변환기)의 개념을 이해한다.

• Verilog를 사용하여 Adder와 Subtractor를 구현한다.

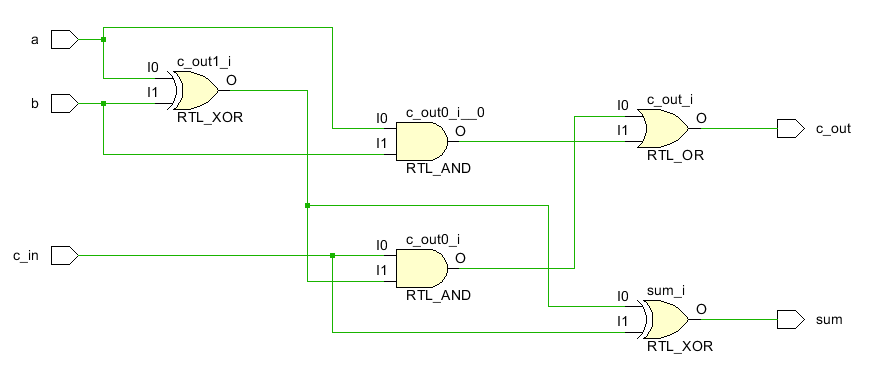
• Verilog를 사용하여 Code converter를 구현한다.

• FPGA를 통해 Verilog로 구현된 회로의 동작을 확인한다.

**2. Full Adder / Half Adder**

Full Adder의 입력 변수는 일반 변수 A, B와 이전 Carry 변수 Cin이고, 출력 변수는 연산 합 S와 Carry 변수 Cout이다. 이들을 논리식으로 표현하면 다음과 같다.

이를 통해 구현한 Schematic, Simulation, Truth Table의 결과는 다음과 같다.



스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** |  |  | **S** |
| **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** |

Half Adder의 입력 변수는 일반 변수 A와 B이고, 출력 변수는 연산 합 S와 Carry 변수 C다. 이들을 논리식으로 표현하면 다음과 같다.

이를 통해 구현한 Schematic, Simulation, Truth Table의 결과는 다음과 같다.

도표, 평면도, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **S** |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **0** |

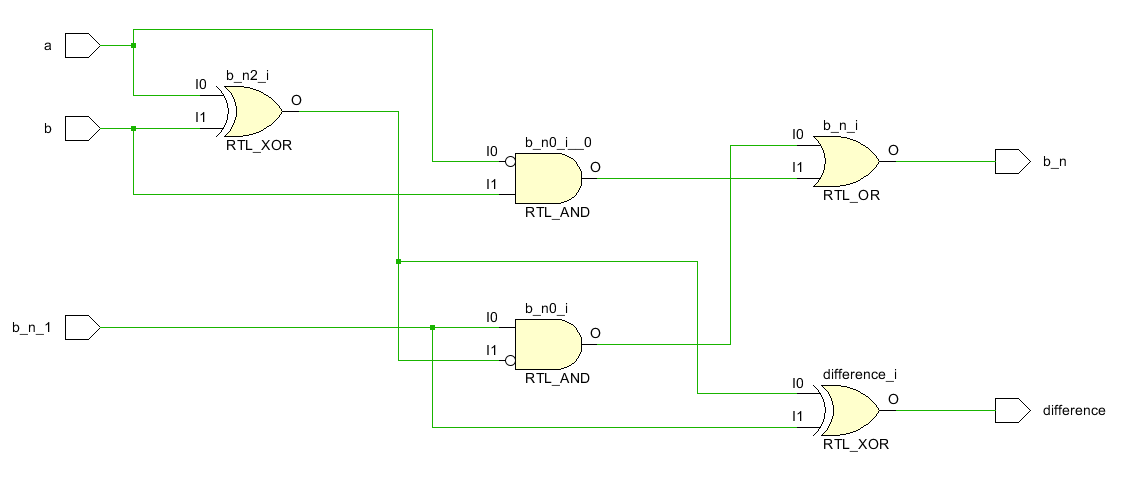
Full Adder에서 S는 A, B, Cin 중에서 1값이 홀수 개이면 그 값이 1이 출력된다. Cout은 입력 값이 모두 1이거나, 입력 값 중 둘 중 하나만 1이고 Cin 값이 1인 경우 1을 출력한다.

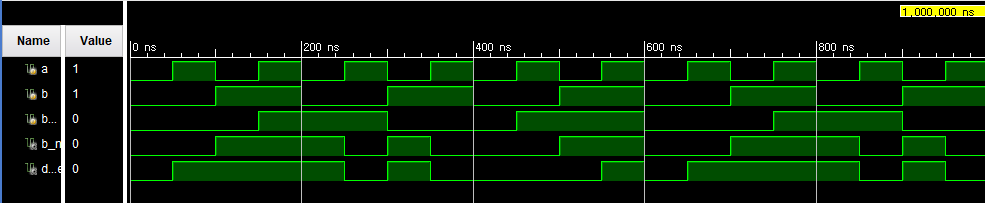
Half Adder에서 S는 두 입력 값 중 하나만 1인 경우 1을 출력한다. Carry는 두 입력 값이 모두 1인 경우 1을 출력한다. 자릿수가 증가했기 때문이다.

**3. Full Subtractor / Half Subtractor**

Full Subtractor의 입력 변수는 일반 변수 A, B와 이전 Borrow 변수 이고, 출력 변수는 연산 차 D와 Borrow 변수 이다. 이들을 논리식으로 표현하면 다음과 같다.

이를 통해 구현한 Schematic, Simulation, Truth Table의 결과는 다음과 같다.





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** |  |  | **D** |
| **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** | **1** |
| **1** | **0** | **1** | **0** | **0** |
| **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** |

Half Subtractor의 입력 변수는 일반 변수 A와 B이고, 출력 변수는 연산 차 D와 Borrow 변수 b다. 이들을 논리식으로 표현하면 다음과 같다.

이를 통해 구현한 Schematic, Simulation, Truth Table의 결과는 다음과 같다.

도표이(가) 표시된 사진

자동 생성된 설명

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **b** | **D** |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** |
| **1** | **1** | **0** | **0** |

Full Subtracter에서 D는 A, B, bn-1 중에서 1값이 홀수 개이면 1을 출력한다. bn은 두 입력 값이 0 또는 1로 동일하면서 bn-1이 1이거나, 또는 A가 0이고 B가 1일 때 1을 출력한다.

Half Subtracter에서 D는 두 입력 값이 서로 다른 경우 1을 출력한다. Borrow는 A가 0이고 B가 1인 경우 1을 출력한다. 자릿수가 감소했기 때문이다.

**4. 8421(BCD)-2421 Code converter**

8421(BCD)-2421의 Truth Table은 다음과 같다.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** |

이를 카르노 맵으로 구현하면 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A2** | | | | | |
|  | **C1D1** | **00** | **01** | **11** | **10** |
| **A1B1** |  |  |  |  |  |
| **00** |  | **0** | **0** | **0** | **0** |
| **01** |  | **0** | **1** | **1** | **1** |
| **11** |  | **X** | **X** | **X** | **X** |
| **10** |  | **1** | **1** | **X** | **X** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **B2** | | | | | |
|  | **C1D1** | **00** | **01** | **11** | **10** |
| **A1B1** |  |  |  |  |  |
| **00** |  | **0** | **0** | **0** | **0** |
| **01** |  | **1** | **0** | **1** | **1** |
| **11** |  | **X** | **X** | **X** | **X** |
| **10** |  | **1** | **1** | **X** | **X** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **C2** | | | | | |
|  | **C1D1** | **00** | **01** | **11** | **10** |
| **A1B1** |  |  |  |  |  |
| **00** |  | **0** | **0** | **1** | **1** |
| **01** |  | **0** | **1** | **0** | **0** |
| **11** |  | **X** | **X** | **X** | **X** |
| **10** |  | **1** | **1** | **X** | **X** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **D2** | | | | | |
|  | **C1D1** | **00** | **01** | **11** | **10** |
| **A1B1** |  |  |  |  |  |
| **00** |  | **0** | **1** | **1** | **0** |
| **01** |  | **0** | **1** | **1** | **0** |
| **11** |  | **X** | **X** | **X** | **X** |
| **10** |  | **0** | **1** | **X** | **X** |

이를 통해 A2, B2, C2, D2를 각각 SOP form, POS form으로 나타내면 다음과 같다.

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

이를 통해 구현한 Schematic, Simulation의 결과는 다음과 같다.

도표, 라인, 텍스트이(가) 표시된 사진

자동 생성된 설명

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

위의 논리식들을 NAND/NOR로만 표현하면 다음과 같다.

e = NOR( NAND( NAND(b,b) , NOR(c,d) ), a )

f = NOR( NOR( a , NAND( NAND(b,b) , NOR(NAND(d,d) , c)) ) , a )

g = NOR( NOR(a , NAND(NAND(b,b),c)) , NAND(NAND(NAND(b,b),NAND(c,c)) , d) )

h = NAND(d, d) 또는 NOR(d, d)

**5. 결과 검토 및 논의 사항**

카르노 맵을 통해 논리식을 도출하는 과정에서 그 답은 여러 개가 도출된다. 값이 1과 X인 칸을 기준으로 어떻게 묶어서 식을 만들 지는 사람마다 다르기 때문이다. 하지만 서로 같은 진리표를 보고 식을 도출하기에 만들어진 논리식의 결과는 서로 같아야 한다. 즉 Verilog의 Schematic은 다를 수 있지만 Simulation 결과는 같다.

NAND/NOR 게이트만을 사용해 모든 논리식을 표현할 수 있지만, 위의 e, f, g, h와 같이 아무리 cost가 낮은 NAND/NOR 게이트이지만 반복해서 사용하면 총 cost가 다른 게이트들을 혼합하여 사용할 때보다 높은 것을 알 수 있다.

**6. 추가 이론**

NAND/NOR 게이트는 De-Morgan 법칙과 연관이 있다. OR 게이트의 NOT이 NAND 게이트이고, AND 게이트의 NOT이 NOR 게이트다. 이를 논리식으로 표현하면 다음과 같다.

**NOT OR (A • B)' = A' + B' NAND**

**NOT AND (A + B)' = A' • B' NOR**