5주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211522 이름: 김정환

**1.**

.................

De Morgan의 정리의 동작을 Verilog로 구현하고, 이를 Simulation과 FPGA를 통해서 동작을 확인한다. 또한 De Morgan의 정리를 응용한 주어진 Boolean 함수의 동작을 Verilog로 구현하여 이를 Simulation과 FPGA를 통해서 확인한다. 마지막으로 1 Bit 비교기를 주어진 대로 구현하고 동작을 Simulation과 FPGA로 확인한다.

................

**2.**

.......................

De Morgan의 제1법칙은 아래와 같은 방식으로 구현했다.텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

각각 왼쪽이 (A), 오른쪽이 (B)를 구현한 것으로, 이는 각각 (A+B)’, A’\*B’를 나타낸다. 해당 코드들의 Simulation 결과는 아래와 같다.

스크린샷, 멀티미디어 소프트웨어, 사각형이(가) 표시된 사진

자동 생성된 설명

(A)의 Simulation

스크린샷, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

(B)의 Simulation

De Morgan의 제1법칙에 의해 서로 같다고 한 것과 마찬가지로 Simulation의 결과도 같은 것을 확인할 수 있다. 이를 NOR Gate와 비교했을 때 (A)의 구현 시 나타낸 ~(a|b)의 표현은 NOR Gate와 같은 방식으로 표현한 것을 확인할 수 있고 (B)는 같은 결과를 보이므로 (A)와 (B) 모두 NOR Gate와 같은 역할을 수행한다.

다음으로 De Morgan의 제2법칙은 아래와 같은 방식으로 구현했다.

텍스트, 폰트, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

각각 왼쪽이 (A), 오른쪽이 (B)를 구현한 것으로 이는 각각 (A\*B)’, A’+B’를 나타낸다. 해당 코드들의 Simulation 결과는 아래와 같다.

스크린샷, 멀티미디어 소프트웨어, 그래픽 소프트웨어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

(A)의 Simulation

스크린샷이(가) 표시된 사진

자동 생성된 설명

(B)의 Simulation

De Morgan의 제2법칙에 따라 서로 같다고 한 것과 동일하게 같은 Simulation 값을 보이는 것을 확인할 수 있다. 이를 NAND Gate와 비교하면 (A)의 구현 방식이 ~(a&b)로 NAND Gate와 같은 방식으로 표현한 것을 볼 수 있다. (B)도 (A)와 같은 결과를 보이므로 (A)와 (B)는 NAND Gate와 같은 역할을 수행한다.

마지막으로 제1법칙과 제2법칙 모두에 대한 진리표는 아래와 같다.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | A+B | A\*B | A’ | B’ | (A+B)’ | A’\*B’ | (A\*B)’ | A’+B’ |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

.........................

**3.**

.......................

(A’+B’)\*C’와 ((A\*B)+C)’의 구현은 아래와 같다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

왼쪽이 (A)로 (A’+B’)\*C’를 나타내고, 오른쪽이 (B)로 ((A\*B)+C)’를 나타낸다. 해당 코드들의 Simulation 결과는 아래와 같다.

스크린샷, 멀티미디어 소프트웨어, 사각형이(가) 표시된 사진

자동 생성된 설명

(A)의 Simulation

스크린샷, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

(B)의 Simulation

Simulation 결과가 서로 같은 것을 통해서 De Morgan의 법칙을 따른 위의 두 식이 같은 것을 확인할 수 있다. 이들의 진리표는 아래와 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| IN A | IN B | IN C | OUT D |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

다음으로, (A’\*B’)+C’와 ((A+B)\*C)’의 구현은 아래와 같다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

왼쪽이 (A)로 (A’\*B’)+C’를 나타내고, 오른쪽이 (B)로 ((A+B)\*C)’를 나타낸다. 해당 코드들의 Simulation은 아래와 같다.

스크린샷, 사각형, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

A의 Simulation

스크린샷, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

B의 Simulation

Simulation의 결과가 서로 같은 것을 확인할 수 있고 이는 De Morgan의 법칙을 적용한 위의 두 식이 같은 결과를 나타낸다는 것을 보인다. 이들의 진리표는 아래와 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| IN A | IN B | IN C | OUT D |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

.........................

**4.**

.......................

1 Bit 비교기에서는 각각 A=B, A≠B, A>B, A<B를 각각 순서대로 c, d, e, f에 출력하도록 구현했다. 이는 아래와 같이 구현했다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

A=B는 1비트끼리는 서로 다를 때 1을 출력하는 XOR에 NOT을 붙인 XNOR Gate를 이용했고, A≠B는 A=B와 유사하나 반대 결과이므로 XOR을 이용했다. A>B와 A<B는 각각 더 작은 쪽은 0이고 나머지 하나가 1일 때만 true이므로 작은 쪽에 NOT을 씌운 것과 AND 연산을 통해서 둘 다 1일 때만 1이 나오도록 구현했다. 이를 simulation한 결과는 아래와 같다.

스크린샷, 멀티미디어 소프트웨어, 사각형이(가) 표시된 사진

자동 생성된 설명

다음으로 이에 대한 진리표를 작성하면,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IN A | IN B | OUT C | OUT D | OUT E | OUT F |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

.........................

**5.**

.......................

첫 번째 실습에서는 De Morgan의 정리를 각각 제1법칙, 제2법칙에 대해서 성립하는지를 확인하였다. 여기에서 각각 AND와 OR에 대해서 전체에 NOT을 씌웠을 때는 각각 서로 반대로 연산자를 바꾸고 각각에 NOT을 처리한 것과 동일함을 보였다. 이는 논리식을 간소화할 때 이용할 수 있고, 이는 두 번째 실습의 Boolean function의 식들이 같음을 보이는 증거가 된다.

두 번째 실습에서는 첫 번째에서 확인한 De Morgan의 정리를 이용해서 만든 3-Input의 식을 확인한다. De Morgan의 정리에 따라서 같은 것을 Simulation과 FPGA에서 같음으로 확인 가능하다.

마지막으로 1 Bit 비교기는 각각 1 Bit에 대해서 AND와 OR, NOT으로 연산자를 구현하는 실습이었다. 이는 XOR을 이용하거나 NOT, AND를 이용하여 구현했다. 해당 결과는 연산자에 맞게 나오는 것을 통해 잘 구현하였음을 확인했다.

.........................

**6.**

.......................

실습에서는 1 Bit에 대해서만 비교기를 구현하였으나 이를 확장하여 2 Bit나 3 Bit 등 n개의 Bit로 확장할 수 있다. 2 Bit에 대해서 알아보면, 각각 2 Bit씩 존재하면, 총 16가지의 경우의 수를 따져보았을 때, 나온 진리표를 바탕으로 회로를 구성한다. 이 때 나온 값에서 회로를 단순화하는 과정을 거쳐야 하는데, 이 때 전에 보고서에 담았던 카르노 맵을 활용하여 논리식을 간소화할 수 있다. 이렇게 1 Bit 비교기와 2 Bit 비교기를 구현하였으면 이들의 조합으로 더 큰 비교기를 만들 수 있다. 예를 들어 =에 대한 연산을 구현 시에는 더 작은 Bit의 비교기의 = 연산에 대한 출력을 각각 A, B라고 하면 A\*B로 구현하는 등의 방식을 통해서 더 큰 Bit로의 확장이 가능하다.

.........................