5주차 결과보고서

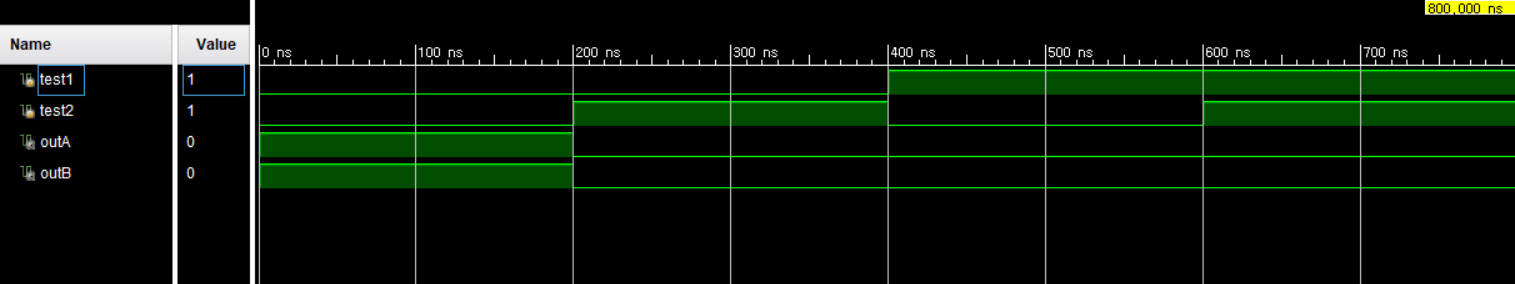
전공: 컴퓨터공학과/생명과학과 학년: 9학기 학번: 20181435 이름: 박다희

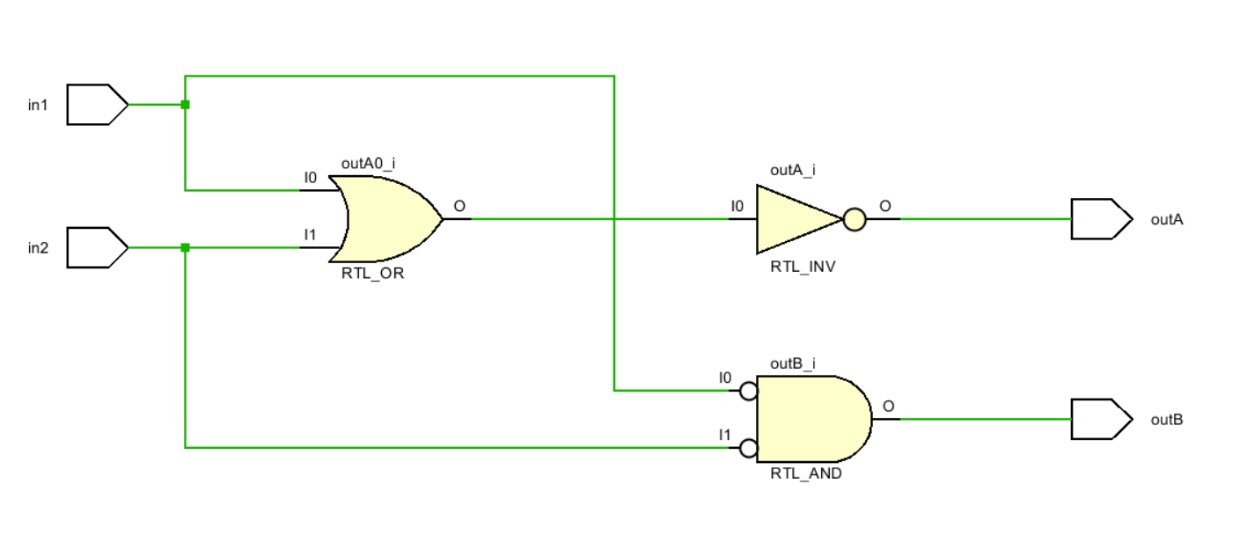
**1.**

이번 실험은 De-Morgan의 제 1법칙, 제 2법칙에 대해서 배우고 Verilog로 De-Morgan의 제 1,2법칙의 동작을 구현해 본 후 simulation의 결과를 확인하고 FPGA로 회로의 동작을 확인하는 것이다. 또한 Boolean함수인 (A'+B')\*C' = ((A\*B)+C)'와 (A'\*B')+C' = ((A+B)\*C)'를 Verilog로 구현하고 simulation결과 및 FPGA로 회로의 동작을 확인하고, 1bit단위로 대소비교와 더불어 같은 수인지 다른 수인지 판별하는 1bit비교기를 Verilog로 구현한 뒤 simulation결과와 FPGA로 회로의 동작까지 확인하는 것이다. 이후엔 각각의 진리표도 작성한다.

텍스트이(가) 표시된 사진

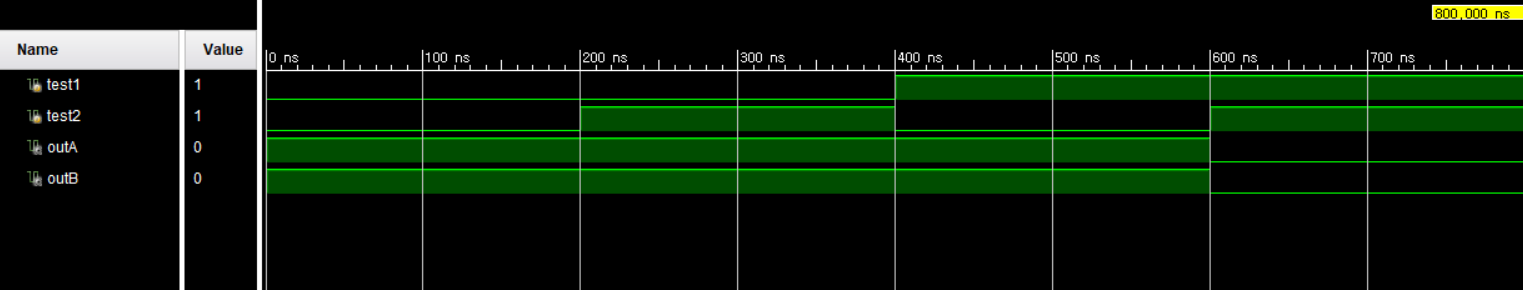
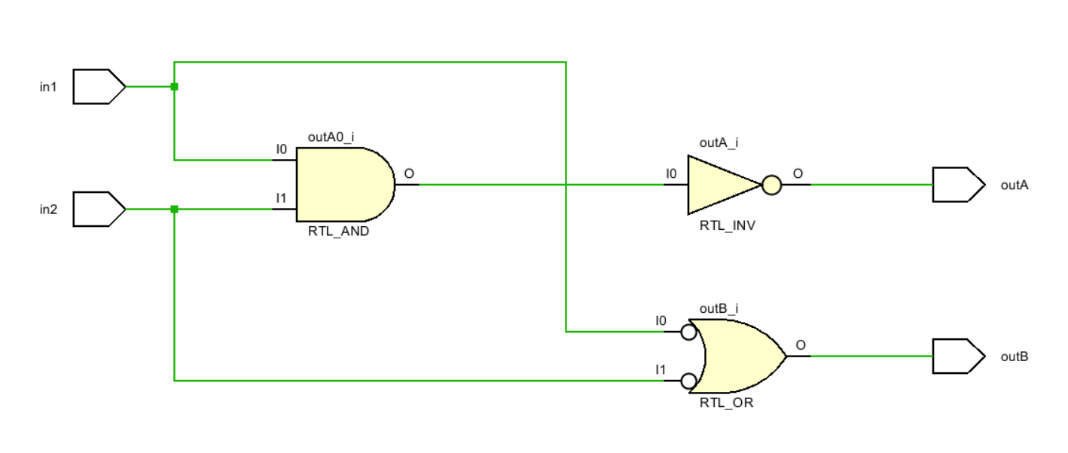
자동 생성된 설명**2.**



 위 그림은 순서대로 testbench, simulation, schematic의 결과이다. 드모르간 제1법칙은 으로 두개의 입력 값을 더한 뒤에 보수를 취한 값과 각각의 보수를 먼저 구하고 후에 논리곱을 취한 값이 같다는 것이다. 이에 따라서 두개의 입력 값에 대해 두개의 출력 값을 (inA+inB)’와 (inA’)(inB’)으로 할당한 schematic의 결과를 보면 outA값은 두개의 입력 값을 OR gate를 통과시킨 후에 NOT gate를 통과시킨다는 것을 확인할 수 있고 outB는 두 입력 값을 먼저 NOT gate에 통과시킨 후에 AND gate를 통과시킨다는 것을 볼 수 있다. 또한 simulation결과를 보면 두 출력 값이 같다는 것을 확인할 수 있다. 이를 통해 NOR연산은 (A+B)’로 나타낼 수 있는데 이 뿐만 아니라 A’B’로도 표현할 수 있다는 것을 알 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명



위 그림은 순서대로 testbench, simulation, schematic의 결과이다. 드모르간 제2법칙은 으로 두개의 입력 값을 논리곱한 뒤에 보수를 취한 값과 각각의 보수를 먼저 구하고 후에 논리합을 취한 값이 같다는 것이다. 이에 따라서 두개의 입력 값에 대해 두개의 출력 값을 {(inA)(inB)}’와 (inA’)+(inB’)으로 할당한 schematic의 결과를 보면 outA값은 두개의 입력 값을 AND gate를 통과시킨 후에 NOT gate를 통과시킨다는 것을 확인할 수 있고 outB는 두 입력 값을 먼저 NOT gate에 통과시킨 후에 OR gate를 통과시킨다는 것을 볼 수 있다. 또한 simulation결과를 보면 두 출력 값이 같다는 것을 확인할 수 있다. 이를 통해 NAND연산은 (AB)’로 나타낼 수 있는데 이 뿐만 아니라 A’+B’로도 표현할 수 있다는 것을 알 수 있다.

**3.**

텍스트, 영수증, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

스크린샷, 사각형이(가) 표시된 사진

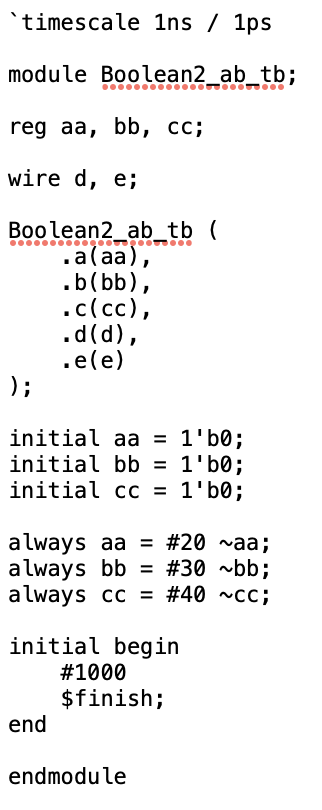
자동 생성된 설명

도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

위의 그림은 차례대로 첫번째 논리함수 (A'+B')\*C' = ((A\*B)+C)'에 대한 testbench, design source, simulation, schematic의 결과이다. 이에 대한 진리표는 다음과 같다.

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

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

도표, 평면도, 라인이(가) 표시된 사진

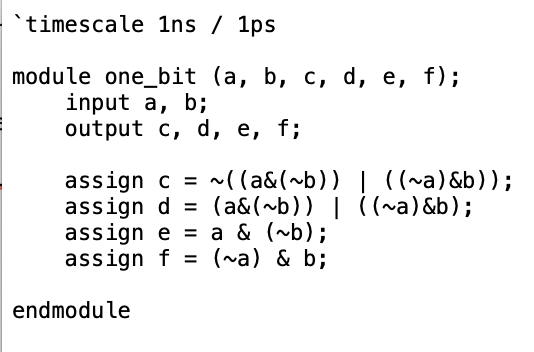
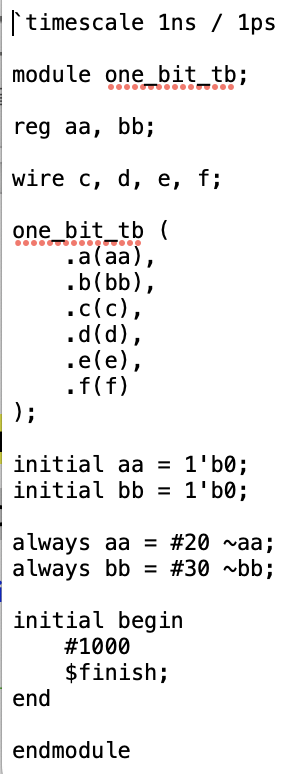
자동 생성된 설명

위의 사진들은 차례대로 두번째 Boolean function인 (A'\*B')+C' = ((A+B)\*C)'에 대한 testbench, design source, simulation, schematic의 결과이다. 이에 대한 진리표는 다음과 같다.

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

위의 2개의 Boolean fuction의 simulation과 진리표 결과를 보면 두 Boolean function은 그 값이 같다는 것을 확인할 수 있다. 이는 앞서 2번에서 설명한 드모르간의 법칙을 사용하여 두 개의 Boolean function이 같다는 것을 보일 수도 있다.

**4.**



스크린샷, 다채로움, 사각형, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

위의 사진들은 차례대로 testbench, design source, simulation, schematic의 결과이다. 이를 바탕으로 진리표를 작성하면 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | A=B | AB | A > B | A < B |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

위의 코드를 작성할 때 연산은 연산으로, 연산은 으로 구현하였고, A=B는 XNOR 연산으로, AB은 XOR 연산으로 구현하였다. 입력 값으로 1아니면 0으로 들어오기 때문에 이러한 대소 비교 연산이 가능한 것이다.

**5.**

이번 실험을 통해 드모르간의 제1, 2법칙을 Verilog로 구현하여 simulation과 진리표를 통해 확인하고 성립하는 것을 볼 수 있었고, 이 후 두개의 Boolean function인 와 , 와 을 Verilog로 구현하고 simulation결과와 진리표를 확인하여 같은 기능을 하는 함수임을 보였다. 이를 통해 드모르간의 법칙을 사용하여 boolean function이 다양한 형태로 나타낼 수 있다는 것을 알 수 있었다. 마지막으론 1-bit 비교기도 verilog로 구현함으로 대소비교와 등위연산을 하였다.

**6.**

SOP(Sum Of Product)로 boolean식을 구성하고 있는 요소들을 곱한 뒤에 이 항들을 다 더한 것을 의미하고, 반대로 POS(Product Of Sum)은 먼저 다 더한 뒤에 이 항들을 다 곱한 것을 의미한다. 이 중에서도 각각의 모든 항의 모든 종류의 요소들이 들어가 있는 것을 canonical form이라고 일컫는다. 주어진 진리표를 통해 canonical form인 SOP 또는 POS를 이용하여 boolean식을 알아낼 수 있다. 만약 다음과 같은 진리표가 있다 하자.

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

F가 1을 갖는 것 만을 골라서 곱한 후에 그 항들을 다 더하면 다음과 같은 식이 된다.

F = A’BC + AB’C + ABC’ + ABC 이 식은 F의 SOP이고, 이제 POS를 구하기 위해서는 F가 0을 갖는 것 만을 골라서 더한 이후에 입력 값이 1인 경우는 부정의 형태로 그 항들을 다 곱하거나 F’의 SOP을 구한 뒤에 드모르간 법칙을 이용하여 구해주면 된다. 예시로는 드모르간 법칙을 사용하는 방법을 택하자면 다음과 같다.

F’ = A’B’C’ + A’B’C + A’BC’ + AB’C’이고 이 식의 전체에 NOT을 취하면 (F’)’ = (A’B’C’ + A’B’C + A’BC’ + AB’C’)’ = (A+B+C)(A+B+C’)(A+B’+C)(A’+B+C)이 된다.