4주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20191599 이름: 송경호

**1.**

• NAND/NOR/XOR/AOI의 gate의 동작을 이해하고 Verilog 코드를 작성한다.

• 작성한 코드들을 바탕으로 simulation을 진행하여 각 gate의 동작을 확인한다.

• FPGA를 연결하여 gate의 동작을 확인한다.

**2.**

**2-1) Diagram**

구현해야 할 4 input 3 output NAND의 diagram이다.

**스케치, 도표, 그림, 화이트이(가) 표시된 사진

자동 생성된 설명**

**2-2) simulation 과정**

(1) 텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명 (2) 텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트, 대수학이(가) 표시된 사진

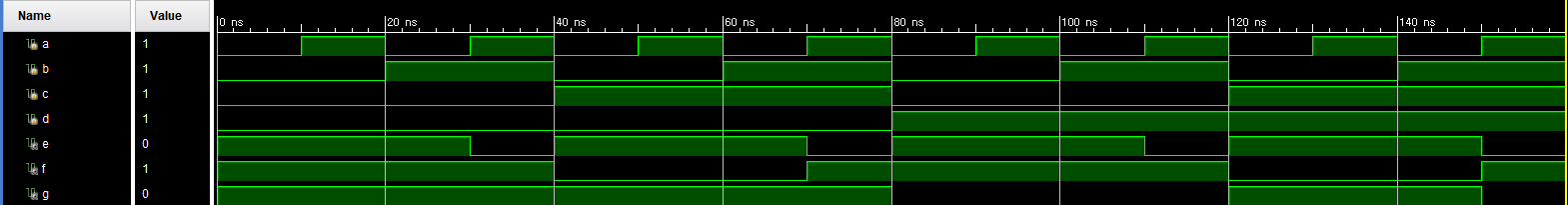
자동 생성된 설명

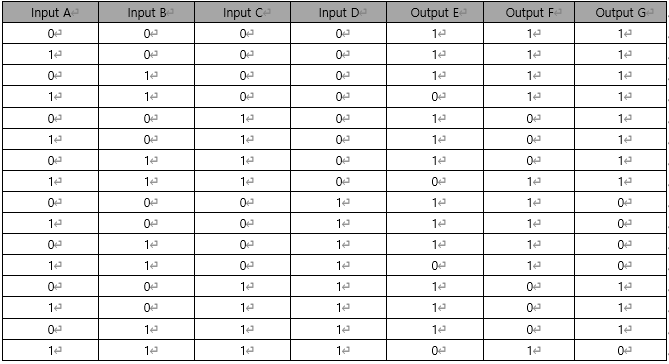
(1)과 (2)가 각 각 4 input 3 ouput NAND를 구현한 디자인 코드와 테스트벤치 코드이다. 위의 diagram을 구현하기 위해

|  |
| --- |
| *assign e = ~(a&&b);*  *assign f = ~(c&&e);*  *assign g = ~g(d&&f);* |

로 디자인 코드를 작성한다. 두 input a, b에 and 연산을 한 것에 not 연산을 한 결과를 다시 input으로 쓰는 방식이다. 또한 4개의 input의 상태를 확인 보기 위해 a, b, c, d를 각 각 10, 20, 40, 80 ns마다 변화시킨다. 결과적으로 160ns에 모든 입력값에 대한 NAND의 결과값을 확인할 수 있다.

**2-3) simulation 결과와 진리표**

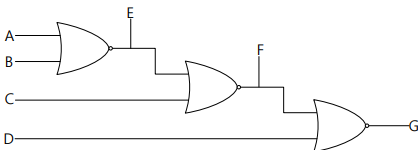
****



**3.**

**3-1) Diagram**

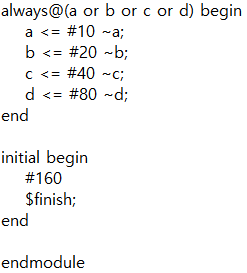
구현해야 할 4 input 3 output NOR의 diagram이다.

****

**3-2) simulation 과정**

(1) 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 (2) 텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

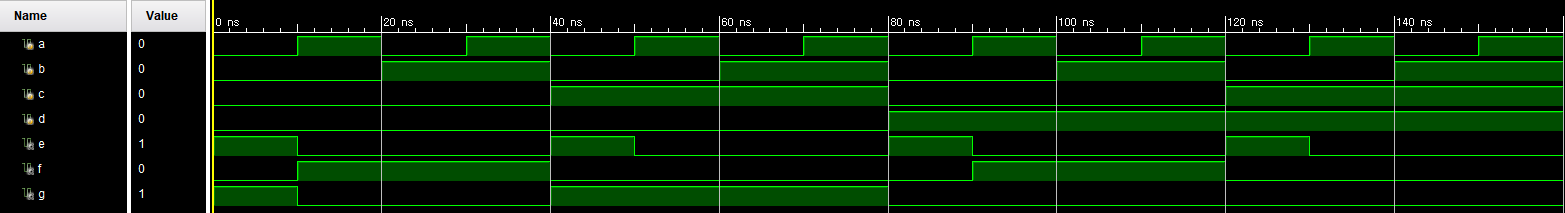
자동 생성된 설명 

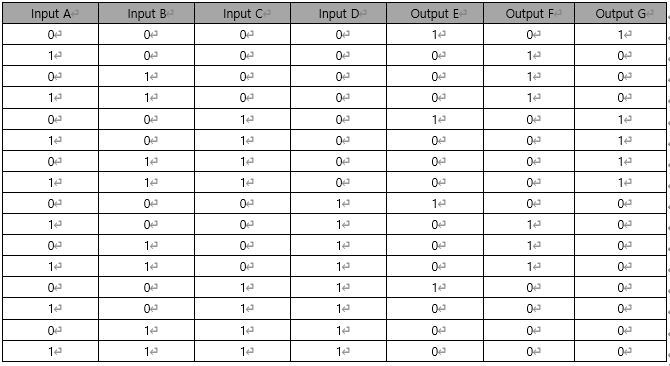
(1)과 (2)가 각 각 4 input 3 ouput NOR를 구현한 디자인 코드와 테스트벤치 코드이다. 위의 diagram을 구현하기 위해

|  |
| --- |
| *assign e = ~(a||b);*  *assign f = ~(c||e);*  *assign g = ~g(d||f);* |

로 디자인 코드를 작성한다. 두 input a, b에 or 연산을 한 것에 not 연산을 한 결과를 다시 input으로 쓰는 방식이다. 또한 4개의 input의 상태를 확인 보기 위해 a, b, c, d를 각 각 10, 20, 40, 80 ns마다 변화시킨다. 결과적으로 160ns에 모든 입력값에 대한 NOR의 결과값을 확인할 수 있다.

**3-3) simulation 결과와 진리표**

****



**4.**

**4-1) Diagram**

구현해야 할 4 input 3 output XOR의 diagram이다.

**도표, 스케치, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명**

**4-2) simulation 과정**

(1) 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 (2) 텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

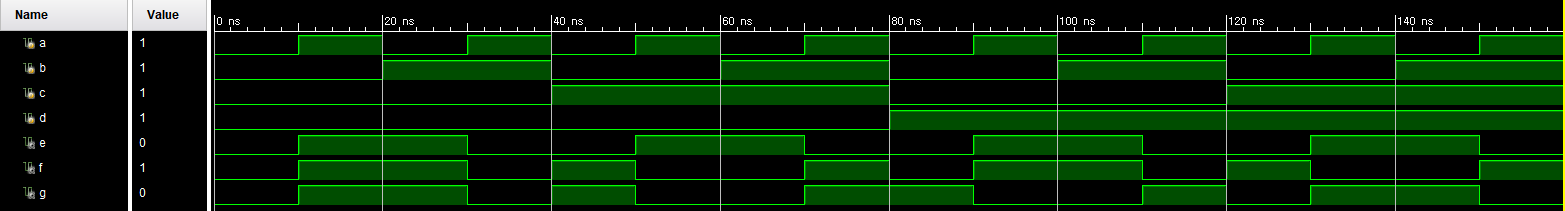
자동 생성된 설명

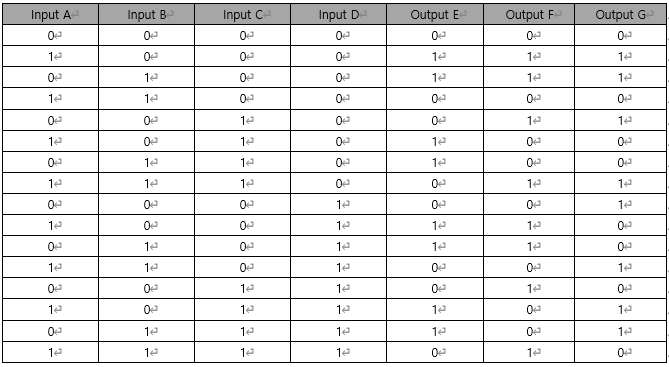
(1)과 (2)가 각 각 4 input 3 ouput XOR를 구현한 디자인 코드와 테스트벤치 코드이다. 위의 diagram을 구현하기 위해

|  |
| --- |
| *assign e =a^b;*  *assign f = c^e;*  *assign g = d^f;* |

로 디자인 코드를 작성한다. 두 input a, b에 xor 연산을 한 결과를 다시 input으로 쓰는 방식이다. 또한 4개의 input의 상태를 확인 보기 위해 a, b, c, d를 각 각 10, 20, 40, 80 ns마다 변화시킨다. 결과적으로 160ns에 모든 입력값에 대한 XOR의 결과값을 확인할 수 있다.

**4-3) simulation 결과와 진리표**

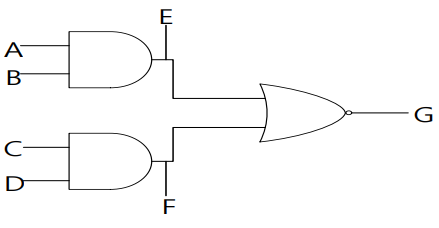
****



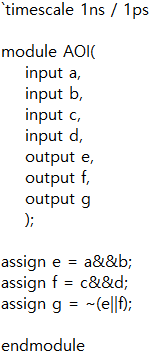
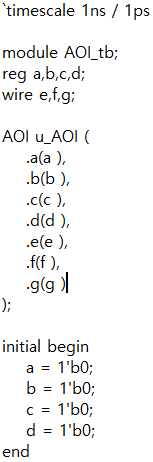
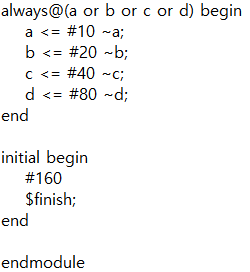
**5.**

**5-1) Diagram**

구현해야 할 4 input 3 output AOI의 diagram이다.

****

**5-2) simulation 과정**

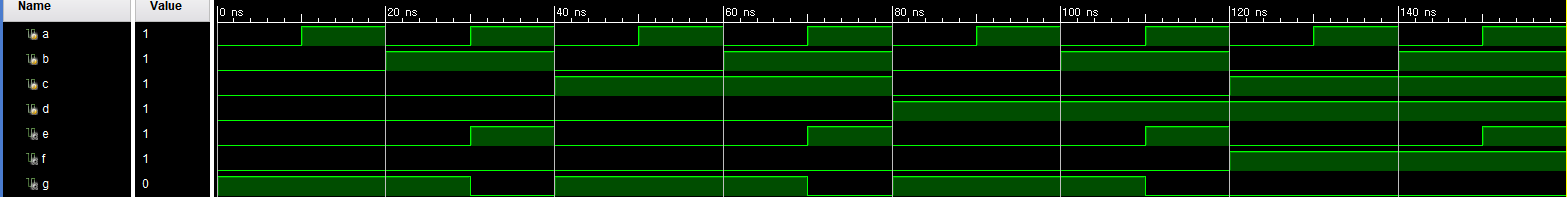
(1)  (2)  

(1)과 (2)가 각 각 4 input 3 ouput AOI를 구현한 디자인 코드와 테스트벤치 코드이다. 위의 diagram을 구현하기 위해

|  |
| --- |
| *assign e = a&&b;*  *assign f = c&&d;*  *assign g = ~(e||f);* |

로 디자인 코드를 작성한다. 위의 세 경우와 달리 먼저 input들에 대한 and연산을 수행 후 두 연산의 결과에 대한 nor연산을 수행한다. nor연산은 or 연산 후에 not 연산을 하는 방식으로 구현했다. 또한 4개의 input의 상태를 확인 보기 위해 a, b, c, d를 각 각 10, 20, 40, 80 ns마다 변화시킨다. 결과적으로 160ns에 모든 입력값에 대한 AOI의 결과값을 확인할 수 있다.

**5-3) simulation 결과와 진리표**

****

텍스트, 번호, 평행이(가) 표시된 사진

자동 생성된 설명

**6.**

**6-1) NAND**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **~(A && B)** |
| **0** | **0** | **1** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

위는 NAND에 두 개의 input이 있을 때의 진리표이다. NAND의 경우 AND에 NOT을 추가한 연산이므로 두 입력값이 모두 1일 때만 0을, 그 외의 경우에는 모두 1을 출력한다. 이번 실험의 4 input 3 output NAND의 경우 하나의 출력값이 다른 출력의 입력값으로 작용하므로 모든 입력값에 대하여 최소 하나 이상의 출력값은 1이라는 것을 알 수 있다. 예를 들어 입력값이 1, 1, 1, 1일 경우 예상과 달리 Output F에는 1이 출력되는데 이는 Output F는 Output E를 입력값으로 사용하기 때문이다.

**6-2) NOR**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **~(A || B)** |
| **0** | **0** | **1** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **0** |

위는 NOR에 두 개의 input이 있을 때의 진리표이다. NOR의 경우 OR에 NOT을 추가한 연산이므로 두 입력값이 모두 0일 때만 1을, 그 외의 경우에는 모두 0을 출력한다. 이번 실험의 4 input 3 output NOR의 경우 입력값에 1이 존재한다고 해서 반드시 모든 출력값이 0이 되는 것은 아니다.

이는 하나의 출력값이 다른 출력의 입력값으로 작용하기 때문에 그런 것으로 예를 들어 입력값이 1, 0, 0, 0일 경우 Output E를 입력값으로 받는 Output F는 1이 출력된다. 그러나 1, 1, 1, 1의 경우 NAND와는 달리 모든 출력값이 예상대로 모두 0으로 출력된다.

**6-3) XOR**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A^B** |
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

위는 XOR에 두 개의 input이 있을 때의 진리표이다. XOR의 경우 두 입력값이 다를 때 1을 같을 때 0을 출력한다. 이번 실험의 4 input 3 output XOR의 경우 모든 입력이 같다고 해서 반드시 모든 출력값이 0이 되지 않았다. 예를 들어 입력값이 1, 1, 1, 1인 경우에도 Output F는 1로 출력 됐는데 이는 Output F가 Output E를 입력값으로 받기 때문이다. 그러나 0, 0 ,0 ,0의 경우에는 모든 Output이 예상대로 0으로 출력된다.

**6-4) AOI**

AOI는 강의 자료를 바탕으로 AND gate 2개의 결과를 NOR gate의 입력값으로 사용하여 구성했다. AND의 출력값이 NOR gate의 입력으로 사용되며 NOR gate의 경우 두 입력값이 0인 경우에만 출력값이 0이 되므로 AND의 입력값이 모두 1인 경우는 반드시 0으로 출력된다.

**7.**

**7-1) XOR의 다른 표현**

실습의 source code에서는 XOR 연산을 A^B와 같이 ‘^’를 통해 구현했다. 그러나 XOR의 실제 논리식은 로 를 통해 나타낼 수 있다. 이를 실제 코드에 적용한다면 X^Y 대신에 (~X&&Y) || (X&&Y)를 통해 구현할 수 있다.

**7-2) XNOR**

실험에서 진행한 AND, OR, NOT, NAND, NOR, XOR, AOI gate 이외에도 XNOR gate도 존재한다. XNOR은 exclusive NOR을 의미하며 NAND와 NOR과 같이 XOR에 NOT을 붙여 구성한다. XOR과 반대로 입력값이 같을 때는 1을, 입력값이 다를 때는 0을 출력한다. 다음은 2 input XNOR gate이다.

도표, 스케치, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

diagram에서 역시 XOR gate에 NOT을 붙인 것을 확인할 수 있다.