3주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20212022 이름: 이예준

**1.**

1. Verilog Coding

vivado 프로그램을 열고 새 프로젝트를 열어 파일명, 저장 위치, 사용할 보드 등을 설정한다.

실행할 코드를 verilog 언어로 작성한다. 코드는 실행에 필요한 design code와 design code를 동작 검증하는 simulation을 위해 필요한 testbench code가 있다.

2. Run Synthesis

작성한 코드를 gate-level netlist로 변환하는 과정을 합성(Synthesis)라고 한다.

이를 통해 하이 레벨의 코드를 로우 레벨의 코드로 변환하여 P&R까지 수행하는 것이 목적이다.

3. Device/Pin Assignment

Device Assignment는 작성한 코드를 물리적으로 실행할 보드를 설정하는 과정이며,

Pin Assignment는 작성한 코드내에 있는 input과 output부분을

보드내에 있는 Switch와 LED에 할당하는 과정이다.

4. Implementation

FPGA 보드에 실제로 디지털 회로를 배치하고 배선하는 과정이며, 이를 Place&Route라고 한다. 이후 FPGA Port로 스트리밍하여 프로그래밍을 하는 bitstream 파일을 생성해준다.

5. Device Configuration

마지막으로 이 과정을 통해 사용자가 설계한 프로그램이 정확히 작동하는지 보드를 통해 확인할 수 있다.

**2.**

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

\*3-input AND gate의 simulation 결과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **In A** | **In B** | **In C** | **Out D** | **Out E** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

\*3-input AND gate 진리표

입력은 A,B,C가 있고, 출력은 D,E가 있다.

출력 D는 입력 A, B를 가지며, 출력 E는 입력 C와 D를 갖는다.

따라서 입력 A와 B가 모두 1일 때에만 출력 D는 1을 갖게 되고,

출력 E 역시 입력 E와 D가 모두 1일 때에만 출력 1을 갖게 된다.

**3.**

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

자동 생성된 설명

\*4-input AND gate simulation 결과

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **In A** | **In B** | **In C** | **In D** | **Out E** | **Out F** | **Out G** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

\*4-input AND gate 진리표

입력은 A, B, C, D가 있고, 출력은 E, F, G가 있다.

출력 E는 입력 A,B를 가지며, 출력 F는 입력 C, E를 가지고 있고

출력 G는 입력 D, F를 가진다. 따라서 입력 A와 B가 모두 1일 때에만 출력 E는 1을

갖게 되고, 입력 C와 E가 모두 1일 때에만 출력 F는 1을 갖게 된다.

마지막으로 출력 G 역시 입력 D와 F가 모두 1일 때에만 1을 갖게 된다

**4.**

스크린샷이(가) 표시된 사진

자동 생성된 설명

\*3-input OR gate simulation 결과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **In A** | **In B** | **In C** | **Out D** | **Out E** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

\*3-input OR gate 진리표

입력은 A,B,C가 있고, 출력은 D,E가 있다.

출력 D는 입력 A, B를 가지며, 출력 E는 입력 C와 D를 갖는다.

따라서 입력 A와 B가 적어도 하나가 1일 때에 출력 D는 1을 갖게 되고,

출력 E 역시 입력 E와 D가 적어도 하나가 1일 때에 출력 1을 갖게 된다.

**5.**

스크린샷, 소프트웨어, 멀티미디어 소프트웨어, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

\* 4-input OR gate simulation 결과

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **In A** | **In B** | **In C** | **In D** | **Out E** | **Out F** | **Out G** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

\*4-input OR gate진리표

입력은 A, B, C, D가 있고, 출력은 E, F, G가 있다. 출력 E는 입력 A,B를 가지며,

출력 F는 입력 C, E를 가지고 있고 출력 G는 입력 D, F를 가진다. 따라서 입력 A와 B가 적어도 하나가 1일 때에 출력 E는 1을 갖게 되고, 입력 C와 E가 적어도 하나가 1일 때에

출력 F는 1을 갖게 된다. 마지막으로 출력 G 역시 입력 D와 F가 적어도 하나가 1일 때에

1을 갖게 된다

**6.**

2~5번 문항을 시뮬레이션, 진리표를 통해 서로 대조했을 때 모두 정상적으로 작동하는 것을 확인했고, 3,5번 문항의 경우에는 FPGA 보드까지 정상적으로 작동하는 것을 확인했다.

이번 실험의 논의사항으로 verilog 언어와 C언어의 AND, OR연산을 탐구했다.

그 결과 verilog 언어와 C언어의 AND, OR연산자가 서로 같음을 확인했고, 그 논리 또한

같은 구조를 띄고 있다는 것도 실험을 통해 확인했다.

**7.**

한 종류의 Gate로만 기본 Gate(AND, OR, NOT)를 구성할 수 있는 Gate를 Universal Gate라고 하며 종류는 NAND와 NOR이 있다.

또한 위에 나온 모든 Gate를 이용해 기본 Gate 3가지를 모두 구현할 수 있다면 그 집합은

완전집합(Complete Set)이라고 한다.

예를 들어 {AND, NOT}은 이미 AND와 NOT Gate가 포함되어 있고, 둘을 조합해 OR Gate를

만들 수 있으므로 {AND, NOT}은 완전집합이라고 할 수 있다.