3주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231523 이름: 김민정

**1. FPGA 동작법**

FGPA의 동작법은 Verilog coding, Device and Pin assignment, Run synthesis, Implement, 마지막으로 Device Configuration 순으로 이루어진다. Device assignment는 Verilog coding을 하기 전, 초기에 새로운 vivado project를 생성할 때 진행할 수 있다.

Verilog coding은 소프트웨어 수준으로 FPGA가 수행할 행동을 작성하는 과정이다. 여기서 무조건 포함되는 파일은 design source file과 testbench file이다. Device assignment는 앞에서 작성했던 project를 실행할 device를 선택하는 과정이다. 현재 우리 실험에서는 Artix-7(xc7a75tfgg484-1)을 사용하고 있다. Pin assignment는 FPGA에서 어떤 곳을 output, input으로 설정할지 정하는 과정이다. 이는 vivado의 constraints File을 통해 설정할 수 있다. 예시는 3주차 강의 자료의 코드이다. 이는 다음과 같다. set\_property -dict {PACKAGE\_PIN G21 IOSTANDARD LVCMOS33} [get\_ports a](33은 pin이 처리할 수 있는 bit이다, 그리고 마지막 a는 design source file에 있는 변수 중 하나가 될 수 있다. 즉 input이나 output이 된다.) 이후 high-level의 code를 FPGA 하드웨어에 적용하기 위해 low-level로 바꾸는 과정을 거치고, input과 output을 각 하드웨어 요소에 적용한다. 이러한 과정을 Run synthesis라고 한다. 이후 Implement 과정에서 만들어진 회로를 효율적으로 실행할 수 있게 요소를 배치한다. 마지막으로 Device Configuration에서는 bitstream file을 FPGA에 업로드하여 위에서 했던 과정이 실행될 수 있게 한다.

**2. 3-input & 2-output AND gate simulation(+source, testbench file)**

|  |  |
| --- | --- |
| Source code | Testbench code |
| 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| simulation |
| 스크린샷, 멀티미디어 소프트웨어, 그래픽 소프트웨어, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 진리표 | | | | |
| A | B | C | D | F |
| 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 |

Input이 모두 1이 되어야 모든 output이 1이 된다.

**3. 4-input & 3-output AND gate simulation(+source, testbench file)**

|  |  |
| --- | --- |
| Source code | Testbench code |
| 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 |

|  |
| --- |
| simulation |
| 스크린샷, 사각형, 그래픽 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진  자동 생성된 설명 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 진리표 | | | | | | |
| A | B | C | D | E | F | 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 |

이도 위와 같이 Input이 모두 1이 되어야 모든 output이 1이 된다.

**4. 3-input & 2-output OR gate simulation(+source, testbench file)**

|  |  |
| --- | --- |
| Source code | Testbench code |
| 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진  자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 |

Testbench code는 3 inputs & 2 output AND gate와 동일하다

|  |
| --- |
| simulation |
| 스크린샷, 사각형, 멀티미디어 소프트웨어이(가) 표시된 사진  자동 생성된 설명 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 진리표 | | | | |
| A | B | C | D | F |
| 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 |

Input이 하나라도 1이면 output은 1이 될 수 있다.

**5. 4-input & 3-output OR gate simulation(+source, testbench file)**

|  |  |
| --- | --- |
| Source code | Testbench code |
| 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 | 텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 |

Testbench code는 4 inputs & 3 output AND gate와 동일하다

|  |
| --- |
| simulation |
| 스크린샷, 사각형이(가) 표시된 사진  자동 생성된 설명 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 진리표 | | | | | | |
| A | B | C | D | E | F | G |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 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 |

Input이 하나라도 1이면 output은 1이 될 수 있다.

**6. 결과 검토 및 논의사항.**

AND gate에서 input과 output의 개수에 상관없이 Input이 모두 1이여야 output은 1이 되는 결과를 가질 수 있었다. 이와 반대로 OR gate에서는 Input 중 하나라도 1이면 output이 1이 되는 결과를 가질 수 있었다.

Logic gate 중에서는 AND, OR뿐만 아니라 NAND, NOR, XOR, XNOR이 더 존재한다. 이는 다음 항목인 7번 문항에서 다뤄보고자 한다.

**7. 추가 조사(NAND, NOR, XOR, XNOR)**

NAND와 NOR은 AND와 OR가 invert가 된 logic gate이다. 즉 결과가 AND와 OR의 정반대가 된다고 생각하면 된다. 아래는 NAND와 NOR의 2-inputs & 1-output의 logic Diagram과 진리표이다.

**가. NAND**

|  |  |  |  |
| --- | --- | --- | --- |
| Logic Diagram | 진리표 | | |
| Input x | Input y | Output z |
|  | 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**나. NOR**

|  |  |  |  |
| --- | --- | --- | --- |
| Logic Diagram | 진리표 | | |
| Input x | Input y | Output z |
|  | 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

XOR은 문제 내에서 하나의 input만이 해당 될 때에만 참이 되는 logic gate이다. 즉 두 개 이상 input 조건 내에 있거나 아무런 input 조건에 해당되지 않는다면 output은 0이 된다. XNOR은 XOR에 inversion이 추가된 것이다. 위에서 말했던 것과 같이 XOR의 결과값에 부정을 취한 것과 같은 결과값이 나오게 된다. 아래는 NXOR와 XOR의 2-inputs & 1-output의 logic Diagram과 진리표이다.

**다. XOR**

|  |  |  |  |
| --- | --- | --- | --- |
| Logic Diagram | 진리표 | | |
| Input x | Input y | Output z |
|  | 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**라. NXOR**

|  |  |  |  |
| --- | --- | --- | --- |
| Logic Diagram | 진리표 | | |
| Input x | Input y | Output z |
|  | 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |