3주차 결과보고서

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

**1.**

FPGA의 동작법은 Vivado라는 프로그램과 Verilog언어를 사용하여 크게 Verilog coding, run synthesis, device/pin assignment, synthesis/implement 그리고 device configuration 이렇게 5단계를 통해 검증할 수 있다. 먼저 Verilog coding은 gate의 동작을 다루는 Verilog design source code와 testbench 코드를 작성하는 단계이다. FPGA에서는 switch를 사용하면 1 또는 0이었던 변수의 값을 다른 값으로 바꿀 수 있다. 예를 들면 switch on은 1이었다면 switch off를 사용하면 0으로 값을 바꿀 수 있다. 다음 단계인 run synthesis 단계는 앞단계에서 작성한 코드를 논리 회로로 변환하는 단계다. 이 단계에서 쓰이는 합성 도구는 Verilog 코드를 논리 gate level로 바꿔주고 논리 회로의 기능을 만든다. 다음 단계인 device/pin assignment는 프로젝트에 적합한 device를 FPGA 동작을 위해 assign하는 단계이다. Vivado에서는 project manager, settings, project device 순의 메뉴에서 설정 가능하고 device는 xc7a75tfgg484-1로 설정해준다. 이 device는 신호 처리나 임베디드 메모리, 로직, 송수신기, LVDS I/O 그리고 메모리 인터페이스 등에 높은 효율과 성능을 제공하는 FPGA device다. Pin은 Verilog 언어에서의 input과 output변수로 나타낼 수 있다. Vivado에서는 Project Manager, Add Sources, Add or create constraints, Create File 순의 메뉴를 통해 .xdc 파일을 만들 수 있고, 이 파일에 할당하고 싶은 FPGA pin list의 Pin과 Verilog 소스의 port를 연결 할 수 있다. 연결할 때는 다음과 같이 작성하면 된다. “set\_property -dict {PACKAGE\_PIN (핀코드) IOSTANDARD LVCMOS33} [get\_ports (변수)]” 그 다음 synthesis/implementation단계는 합성된 회로를 FPGA 칩에 올리는 단게이다. 이 단계에서는 논리 회로와 FPGA 내부와 매핑하는 방법을 결정하고 논리 회로를 물리적인 리소스로 바꿔준다. Vivado에서는 Run Synthesis, Implementation, Run Implementation, Open Implemented Design을 통해 Synthesis Design과 Route Design을 만들 수 있다. 마지막 단계인 device configuration은 FPGA 프로그래밍을 위해 만들어진 design을 FPGA 칩 내부의 구성 메모리에 올리는 단계이다. 이 때 비트스트림 파일을 만들고 이것을 FPGA에 업로드하여 활성화 시킨다. Vivado에서는 Generate Bitstream, Open Hardware Manager를 통해 Device configuration이 이루어진다.   
 이후에는 Open Target, Auto Connect 순으로 메뉴를 선택하고 Project[제목], Project[제목]\_runs, impl\_1 순서의 폴더로 들어가서 debug\_nets.ltx 파일을 추가한 뒤에  
Program and debug, Program Device, Debug probes file 순으로 메뉴를 클릭한 뒤에 debug\_nets.ltx 파일을 선택하여 Program device(xc7a75t\_0)로 외부 FPGA 기기에 연결할 수 있다.

**2.**

**텍스트, 소프트웨어, 컴퓨터 아이콘, 웹 페이지이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명**

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

자동 생성된 설명**

위의 사진들은 차례대로 testbench, design source, simulation결과이다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input A** | **Input B** | **Input C** | **Output D** | **Output 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** |

Input이 3개, output이 2개 이므로 input 변수로는 a, b, c를 output 변수로는 d와 e를 선언하여 a와 b의 AND연산 결과 값을 d에 씌우고, d와 c의 AND 연산 결과 값을 e에 씌웠다. 따라서 output변수 e가 이 논리 회로의 최종 출력 값이고, input 값이 다 1이어야지만 e도 1이 되고 다른 경우에선 0의 값을 가진다.

**3.**

**텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명스크린샷, 사각형, 디자인이(가) 표시된 사진

자동 생성된 설명**

위의 사진들은 차례대로 testbench, design source, simulation결과이다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input A** | **Input B** | **Input C** | **Input D** | **Output E** | **Output F** | **Output 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** |

Input이 4개, output이 3개 이므로 input 변수로는 a, b, c, d를 output 변수로는 e와 f, g를 선언하여 a와 b의 AND연산 결과 값을 e에 씌우고, e와 c의 AND 연산 결과 값을 f에 씌우고 f와 d의 AND연산 결과 값을 g에 씌웠다. 따라서 output변수 g가 이 논리 회로의 최종 출력 값이고, 4개의 input 값이 다 1이어야지만 g도 1이 되고 다른 경우에선 모두 0이다.

**4.**

**텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 컴퓨터, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

위의 사진들은 차례대로 testbench, design source, simulation결과이다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input A** | **Input B** | **Input C** | **Output D** | **Output 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** |

Input이 3개, output이 2개 이므로 input 변수로는 a, b, c를 output 변수로는 d와 e를 선언하여 a와 b의 OR연산 결과 값을 d에 씌우고, d와 c의 OR 연산 결과 값을 e에 씌웠다. 따라서 output변수 e가 이 논리 회로의 최종 출력 값이고, input 값이 다 0이어야지만 e도 0이 되고 다른 경우에선 1의 값을 가진다.

**5.**

**텍스트, 번호, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 컴퓨터, 사각형, 키보드이(가) 표시된 사진

자동 생성된 설명**

위의 사진들은 차례대로 testbench, design source, simulation결과이다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input A** | **Input B** | **Input C** | **Input D** | **Output E** | **Output F** | **Output 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 4개, output이 3개 이므로 input 변수로는 a, b, c, d를 output 변수로는 e와 f, g를 선언하여 a와 b의 OR연산 결과 값을 e에 씌우고, e와 c의 OR 연산 결과 값을 f에 씌우고 f와 d의 OR연산 결과 값을 g에 씌웠다. 따라서 output변수 g가 이 논리 회로의 최종 출력 값이고, 4개의 input 값이 다 0이어야지만 g도 0이 되고 다른 경우에선 모두 1이다.

**6.**

(1) 3-input AND gate (3 input, 2 output)

도표, 라인, 기술 도면, 폰트이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 평면도, 그래프이(가) 표시된 사진

자동 생성된 설명

위의 사진은 강의자료의 회로 그림이고 밑의 사진은 vivado 프로그램의 회로 그림이다. 겉보기에는 vivado 프로그램의 회로 그림이 더 복잡해 보이지만 전체적으로 봤을 때 같은 모습이라는 것을 알 수 있다.

(2) 4-input AND gate (4 input, 3 output)

도표, 라인, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

위의 사진이 강의자료의 회로 그림이고 밑의 사진이 vivado 프로그램의 회로 그림이다. 이 회로도도 앞의 경우와 마찬가지로 겉보기엔 vivado 프로그램의 회로 그림이 더 복잡해 보이지만 전체적으로 같은 모습임을 알 수 있다.

(3) 3-input OR gate (3 input, 2 output)

텍스트, 도끼, 벡터그래픽이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 평면도, 그래프이(가) 표시된 사진

자동 생성된 설명

위의 사진은 강의자료의 회로 그림이고 밑의 사진은 vivado 프로그램의 회로 그림이다. 겉보기에는 vivado 프로그램의 회로 그림이 더 복잡해 보이지만 전체적으로 봤을 때 같은 모습이라는 것을 알 수 있다.

(4) 4-input OR gate (4 input, 3 output)

텍스트, 도끼이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

위의 사진이 강의자료의 회로 그림이고 밑의 사진이 vivado 프로그램의 회로 그림이다. 이 회로도도 앞의 경우와 마찬가지로 겉보기엔 vivado 프로그램의 회로 그림이 더 복잡해 보이지만 전체적으로 같은 모습임을 알 수 있다.

**7.**

(1) gate

Gate는 디지털 시스템 회로의 가장 기본적인 요소로 0과 1로 이루어진 논리 대수에 의한 연산을 집적회로 형태로 구현한 것이다. 기본 gate로는 AND, OR, NOT 등이 있고 이 gate들을 조합하여 여러 다양한 gate들을 만들어 낼 수 있다.

(2) AND gate

AND gate는 기본 디지털 논리 gate로 논리 곱을 나타낸다. 만약 AND gate를 사용할 수 없다면 NAND 또는 NOR gate를 사용하여 AND gate의 역할을 수행하는 회로를 구현해낼 수 있다.

(3) OR gate

OR gate는 기본 디지털 논리 gate로 논리 합을 나타낸다. OR gate를 이용할 수 없다면 AND gate와 마찬가지로 NAND 또는 NOR gate를 사용하여 OR gate 역할을 수행하는 회로를 구현해낼 수 있다.

(4) NOT gate / buffer gate

Inverter라고도 불리는 NOT gate는 출력 값으로 입력값을 반전시켜 내보내고, NOT gate를 두 개 이어붙이면 buffer gate가 된다. 입력 값을 바꾸진 않지만 세기를 강화시킨다.

(4) Boolean식

식을 통해 논리 연산자를 나타내는 식을 논리식이라고 한다. 논리 식이 복잡하더라도 gate 식의 법칙과 연산을 이용하여 보다 간결한 gate의 연결로 나타낼 수 있다. 또한 진리표를 사용하면 논리식의 결과를 추출해낼 수 있다. Boolean식의 법칙으로는 항등원 법칙, 보수 법칙, 등역 법칙, 경계 법칙, 대합 법칙, 교환 법칙, 연관 법칙, 분배 법칙, 드모르간 법칙, 흡수 법칙 등이 있다.