3주차 결과보고서

전공: 아트엔테크놀로지 학년: 3학년 학번: 20191172 이름: 함승우

**1. FPGA 동작법을 설명하시오.**

첫번째로 FPGA 보드와 컴퓨터를 USB를 사용해서 연결해야 한다.

그 후 Verilog 문법에 따라 Vivado HDL을 사용하여 설계한 회로의 design source를 v파일에, simulation할 때 필요한 testbench(줄여서 tb파일이라고 하기도 한다)를 코딩을 한다. 이전에 프로젝트를 생성할 때 project deice에서 xc7a75tfgg484-1를 설정했는데 이와 마찬가지로 project device가 이미 설정한 바와 같은 것인지 확인한다. 그후 constraint를 만든다.

이 xdc파일은 설계보드에 Pin number를 활용하여 어떻게 입력을 줄 것인지 (1, 0), 출력을 LED로 할지 등을 정하는 구문을 쓰는데 예를 들어보면

*set\_property -dict {PACKAGE\_PIN G21 IOSTANDARD LVCMOS33} [get\_ports a]*

*set\_property -dict {PACKAGE\_PIN F15 IOSTANDARD LVCMOS33} [get\_ports y]*

이 두 문장에서 첫번째 문장은 입력값인 a는 G21(PUSH\_SW\_MID) FPGA 가운데 버튼을 의미하며, y는 F15(LED<0>)으로 출력을 나타나게 한다는 것을 알 수 있다. 실습 과정처럼 여러가지의 입력, 여러가지의 출력이 나올 경우에는 위 두 문장을 비롯해 다른 스위치, 다른 LED로 구문을 구성해야 한다.

마지막으로 Run Synthesis와 Run implement를 Vivado상에서 누르고, Generate Bitstream을 클릭한다.

**2. 3-input AND gate의 simulation 결과 및 과정에 대해서 설명하시오.**

3-input, 2output AND gate는 A, B, C라는 3개의 input을 가지고 2개의 output을 가지는 논리 게이트이다. 예비보고서에 작성했듯이 AND 게이트는 두 개 이상의 입력 값을 가지고 하나의 출력값을 출력하는 논리 게이트이다. 따라서, (B)(A, B, C가 순차적으로 입력되는 경우에는 )A와 B로 D라는 output이 출력되고, 이 D라는 출력 값이 C와 함께 input으로 받아 최종적으로 E를 출력한다.

하지만 (A)(Input A, B, C가 한번에 입력되는 경우)에는 바로 D라는 출력이 나온다.

도표, 라인, 화이트, 디자인이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| Inv.v | Inv\_tb.v |
| `timescale 1ns / 1ps  inv(  input a, b, c,  output d  );  assign d = a & b &c;  endmodule | `timescale 1ns / 1ps  module inv\_tb;  reg aa, bb, cc;  wire d;  inv u\_inv (  .a (aa),  .b (bb),  .c(cc),  .d (d ) );  initial aa = 1’b0;  initial bb = 1’b0;  initial cc = 1’b0;  always aa = #100 ~aa;  always bb = #50 ~bb;  always cc = #25 ~cc;  initial begin  #1000  finish  end  endmodule |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input A | Input B | Input 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 |

<(A)의 경우>

도표, 라인, 폰트, 평면도이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  inv(  input a, b, c,  output d, e  );  assign d = a & b;  assign e = d & c;  endmodule | `timescale 1ns / 1ps  module inv\_tb;  reg aa, bb, cc;  wire d, e;  inv u\_inv (  .a (aa),  .b (bb),  .c(cc),  .d (d ), .e(e ) );  initial aa = 1’b0;  initial bb = 1’b0;  initial cc = 1’b0;  always aa = #100 ~aa;  always bb = #50 ~bb;  always cc = #25 ~cc;  initial begin  #1000  finish  end  endmodule |

진리표를 작성하자면 아래와 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input A | Input B | Input 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 |

<(B)의 경우>

위에서 볼 수 있듯이 3가지의 input으로 and 연산을 했을 경우에 진리표의 결과는 똑같이 나온다.

**3. 4-input AND gate의 simulation 결과 및 과정에 대해서 설명하시오.**

4-input AND gate는 a,b,c,d의 4가지 input 값에 대하여 3가지의 output e,f,g가 나오는 AND 게이트이다. (B)는 위와 마찬가지로 a와 b로 e가 출력되고, e와 c를 입력 받아, f를 출력하고, f와 d가 입력 받아 최종적으로 G가 출력된다. (A)의 경우에는 A, B, C, D가 한번에 입력 받는 경우이고, E가 한번에

도표, 라인, 스케치, 디자인이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  inv(  input a, b, c, d,  output e  );  assign e = a & b & c & d;  endmodule | `timescale 1ns / 1ps  module inv\_tb;  reg aa, bb, cc, dd;  wire e;  inv u\_inv (  .a (aa),  .b (bb),  .c(cc),  .d(dd),  .e(e ) );  initial aa = 1’b0;  initial bb = 1’b0;  initial cc = 1’b0;  initial dd = 1’b0;  always aa = #100 ~aa;  always bb = #50 ~bb;  always cc = #25 ~cc;  always dd = #12 ~ dd;  initial begin  #1000  finish  end  endmodule |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input A | Input B | Input C | Input D | Output E |
| 0 | 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 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

<(A)의 경우>

도표, 라인, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  inv(  input a, b, c, d,  output e,f,g  );  assign e = a& b;  assign f = e & c;  assign g = f & d;  endmodule | `timescale 1ns / 1ps  module inv\_tb;  reg aa, bb, cc, dd;  wire e, f, g;  inv u\_inv (  .a (aa),  .b (bb),  .c(cc),  .d(dd),  .e(e ), .f(f ), .g(g ) );  initial aa = 1’b0;  initial bb = 1’b0;  initial cc = 1’b0;  initial dd = 1’b0;  always aa = #100 ~aa;  always bb = #50 ~bb;  always cc = #25 ~cc;  always dd = #12 ~ dd;  initial begin  #1000  finish  end  endmodule |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |

<(B)의 경우>

이것 또한 마찬가지로 B는 A와 다르게 중간 결과값도 있을 뿐, 근본적인 진리표는 변하지 않는다.

**4. 3-input OR gate의 simulation 결과 및 과정에 대해서 설명하시오.**

OR 게이트 또한 역시 두 가지 이상의 input이 있어야한다. 3가지 변수를 입력할 것인데 (A)는 한꺼번에, (B)는 순차적으로 입력했다.

도표, 라인, 스케치, 디자인이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| Inv.v | Inv\_tb.v |
| `timescale 1ns / 1ps  inv(  input a, b, c,  output d  );  assign d = a | b |c;  endmodule | `timescale 1ns / 1ps  module inv\_tb;  reg aa, bb, cc;  wire d;  inv u\_inv (  .a (aa),  .b (bb),  .c(cc),  .d (d ) );  initial aa = 1’b0;  initial bb = 1’b0;  initial cc = 1’b0;  always aa = #100 ~aa;  always bb = #50 ~bb;  always cc = #25 ~cc;  initial begin  #1000  finish  end  endmodule |

|  |  |  |  |
| --- | --- | --- | --- |
| Input A | Input B | Input C | Output D |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

<(A)의 경우>

도표, 라인, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  inv(  input a, b, c,  output d, e  );  assign d = a | b;  assign e = d | c;  endmodule | `timescale 1ns / 1ps  module inv\_tb;  reg aa, bb, cc;  wire d, e;  inv u\_inv (  .a (aa),  .b (bb),  .c(cc),  .d (d ), .e(e ) );  initial aa = 1’b0;  initial bb = 1’b0;  initial cc = 1’b0;  always aa = #100 ~aa;  always bb = #50 ~bb;  always cc = #25 ~cc;  initial begin  #1000  finish  end  endmodule |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

<(B)의 결과>

(A)의 경우에는 한꺼번에 연결을 하였는데 (B)의 경우에도 한꺼번에 연결을 했는데, 두 경우 중간 단계를 제외하고는 진리표 및 결과가 동일하다.

**5. 4-input OR gate의 simulation 결과 및 과정에 대해서 설명하시오.**

이번에 4가지의 입력 개수가 정해져 있고 한꺼번에 출력하는 case는 (A), 순차적으로 중간단계 거치는 case는 (B)이다.

라인, 도표, 스케치, 디자인이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  inv(  input a, b, c, d,  output e  );  assign e = a | b | c | d;  endmodule | `timescale 1ns / 1ps  module inv\_tb;  reg aa, bb, cc, dd;  wire e;  inv u\_inv (  .a (aa),  .b (bb),  .c(cc),  .d(dd),  .e(e ) );  initial aa = 1’b0;  initial bb = 1’b0;  initial cc = 1’b0;  initial dd = 1’b0;  always aa = #100 ~aa;  always bb = #50 ~bb;  always cc = #25 ~cc;  always dd = #12 ~ dd;  initial begin  #1000  finish  end  endmodule |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input A | Input B | Input C | Input D | Output E |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 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 |

<(A)의 경우>

도표, 라인, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  inv(  input a, b, c, d,  output e,f,g  );  assign e = a | b;  assign f = e | c;  assign g = f | d;  endmodule | `timescale 1ns / 1ps  module inv\_tb;  reg aa, bb, cc, dd;  wire e, f, g;  inv u\_inv (  .a (aa),  .b (bb),  .c(cc),  .d(dd),  .e(e ), .f(f ), .g(g ) );  initial aa = 1’b0;  initial bb = 1’b0;  initial cc = 1’b0;  initial dd = 1’b0;  always aa = #100 ~aa;  always bb = #50 ~bb;  always cc = #25 ~cc;  always dd = #12 ~ dd;  initial begin  #1000  finish  end  endmodule |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 | 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 |

<(B)의 경우>

**6. 결과 검토 및 논의사항.**

3가지의 input을 갖는 and, or 논리회로와, 4가지의 input을 갖는 and, or 논리회로를 각각 살펴봤다. and와 or은 각각 두가지 이상의 입력을 받고 반드시 하나의 출력을 갖는다. 따라서, 3가지, 4가지 input을 한 번에 입력할 수도 있었고, 순차적으로 입력할 수도 있었다. 하지만, 한 번에 입력한 (a)와 중간 결과가 도중에 나오는 (b)는 최종적으로 나오는 결과가 모두 동일함을 알 수 있었다.

**7. 추가 이론 조사 및 작성.**

회로가 복잡해짐에 따라, and, or, not의 gate가 많이 쓰일수록, 사람이 보았을 때 직관적으로 알아보기 힘들 것 같았다. 따라서, Boolean algebra의 법칙들을 몇 가지 사용하여 수식을 간단하게 한 후에 효율적인 논리 회로를 설계할 수 있을 것 같았다.

아래는 Boolean algebra이다.

|  |  |
| --- | --- |
| A + B = B + A  AB = BA | 교환법칙 |
| A + (B + C) = (A + B) + C  A(BC) = (AB)C | 결합법칙 |
| A + 0 = A /Aㆍ1 = A | 항등법칙 |
| A(B + C) = AB + AC  A + BC = (A + B)(A + C) | 분배법칙 |
| A + AB = A  A(A + B) = A | 흡수법칙 |
| (A + B)’ = A’B’  (AB)’ = A’ + B’ | 드모르간의 법칙 |
| AB + AB’ = (A + B)(A + B’) | 인접 법칙 |

Boolean algebra를 학습했고, 3주차 예비보고서를 쓰는 도중, 트랜지스터 레벨을 공부하며 몇가지 트랜지스터에 대하여 학습했다. 트랜지스터 레벨로 and, or, not의 gate를 표현하는 과정에서 n형 반도체, p형 반도체에 따라서 MOSFET을 학습했다. 실제로 논리회로를 만드는 가장 기본적인 단위는 트랜지스터인데, 트랜지스터에 다양한 종류가 있으나, 그 중 대표적인 것으로는 MOSFET과 CMOS가 있다. MOSFET의 구조는 아래 그림과 같다. MOSFET에는 NMOS(n형 반도체), PMOS(p형 반도체)가 있다.

텍스트, 스크린샷, 폰트, 직사각형이(가) 표시된 사진

자동 생성된 설명