**4주차 결과보고서**

전공: 아트&테크놀로지학과 학년: 4학년 학번: 20191048 이름: 김도솔

**1.**

NAND/NOR/XOR Gate가 어떻게 동작하는지 이해하고, 실습에서 각 Gate를 직접 구현해봄으로써 이를 확인한다. 우선 Verilog 언어로 Vivado 프로그램에 다중입력 NAND/NOR/XOR Gate를 코드 상으로 구현한다. 다음으로 입력 신호를 생성한 후 Simulation을 통하여 구현된 각 Gate가 어떻게 동작하는지(어떤 출력값을 내보내는지) 확인한다. 마지막으로 FPGA를 연결해 Verilog로 구현된 회로가 의도한 대로 동작하는지 확인한다.

**2.**

**4-input NAND gate**

도표, 라인, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명

1. Boolean식

4-input NAND gate의 경우 A와 B를 NAND gate에 연결해 E를 출력하고, E와 C를 NAND gate에 연결해 F를 출력하고, F와 D를 NAND gate에 연결해 G를 출력한다. 따라서 (B)의 Boolean식은 , , 의 순서로 구성할 수 있다.

1. Verilog 코딩

위의 Boolean식에 따라 Verilog 코드를 작성한 결과는 다음과 같다.

|  |  |
| --- | --- |
| Design Source Code | Simulation Source Code |
| `timescale 1ns / 1ps  module 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 begin  aa = 1'b0;  bb = 1'b0;  cc = 1'b0;  dd = 1'b0;  end  always begin  aa = #20 ~aa;  bb = #30 ~bb;  cc = #40 ~cc;  dd = #50 ~dd;  end  initial begin  #1000  $finish;  end  endmodule |

1. Design Source Code:

inv 모듈은 4개의 입력(a, b, c, d)과 3개의 출력(e, f, g)을 가지고 있다. e는 a와 b의 AND 연산의 NOT 결과이다. f는 e와 c의 AND 연산의 NOT 결과이다. g는 f와 d의 AND 연산의 NOT 결과이다.

2. Simulation Source Code:

inv\_tb 모듈은 inv 모듈을 테스트하기 위한 테스트벤치(Testbench) 역할을 한다. 먼저 inv 모듈을 인스턴스화하고, 입력 및 출력 신호를 연결한다. 다음으로 초기 블록(initial)에서 입력 신호 aa, bb, cc, dd를 초기화한다. always 블록에서는 입력 신호 aa, bb, cc, dd를 일정한 주기로 반전시킨다. 마지막 initial 블록에서는 1000ns(나노초) 후에 시뮬레이션을 종료하는 $finish 명령을 실행한다.

1. Simulation 결과

스크린샷, 소프트웨어, 텍스트, 디스플레이이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드를 시뮬레이션 돌린 결과이다. 시간에 따라 입력 A, B, C, D의 값을 바꾸면 그에 따라 출력 E, F, G의 값이 변화하는 것을 확인할 수 있다. 출력 E는 A와 B가 둘다 1인 경우에만 0의 값을 가진다. A와 B 둘 중 적어도 하나가 0의 값을 가지면 출력 값으로 1을 내보낸다. 출력 F는 E와 C의 값이 둘다 1인 경우에만 0의 값을 가진다. E와 C 둘 중 적어도 하나가 0의 값을 가지면 출력 값으로 1을 내보낸다. 출력 G는 F와 D의 값이 둘다 1인 경우에만 0의 값을 가진다. F와 D 둘 중 적어도 하나가 0의 값을 가지면 출력 값으로 1을 내보낸다. 이 시뮬레이션 결과를 통해 5)의 진리표를 작성할 수 있다. 작성된 진리표를 통해 다중입력 NAND 게이트의 최종 출력 G는 입력 값 A, B, C, D가 (0001), (0101), (1001), (1101), (1111)인 경우에 0의 값을 가지고, 나머지 경우에는 1의 값을 가진다는 것을 알 수 있다. 이를 통해 이전 주차에서 실습했던 AND 게이트는 다단으로 설계해도 모든 입력이 1이면 1을 출력한다는 AND 게이트의 특성을 유지하지만, NAND 게이트는 다단으로 설계하는 경우 모든 입력이 1인 경우에만 0을 출력한다는 NAND 게이트의 특성을 유지하지 않음을 확인할 수 있었다.

1. Schematic Diagram

도표, 라인, 텍스트, 그래프이(가) 표시된 사진

자동 생성된 설명

위는 Vivado에서 생성한 4-input NAND gate의 Schematic Diagram이다. 입력 A와 B를 NAND(AND+NOT) gate에 연결해 E를 출력하고, 입력 E와 C를 NAND(AND+NOT) gate에 연결해 F를 출력하고, 입력 F와 D를 NAND(AND+NOT) gate에 연결해 G를 출력한다는 점에서 작성한 코드가 (B)와 동일하게 동작하는 것을 알 수 있다.

1. 4-input NAND gate의 진리표

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

**3.**

**4-input NOR gate**

도표, 라인, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명

1. Boolean식

4-input NOR gate의 경우 A와 B를 NOR gate에 연결해 E를 출력하고, E와 C를 NOR gate에 연결해 F를 출력하고, F와 D를 NOR gate에 연결해 G를 출력한다. 따라서 (B)의 Boolean식은 , , 의 순서로 구성할 수 있다.

1. Verilog 코딩

위의 Boolean식에 따라 Verilog 코드를 작성한 결과는 다음과 같다.

|  |  |
| --- | --- |
| Design Source Code | Simulation Source Code |
| `timescale 1ns / 1ps  module 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 begin  aa = 1'b0;  bb = 1'b0;  cc = 1'b0;  dd = 1'b0;  end  always begin  aa = #20 ~aa;  bb = #30 ~bb;  cc = #40 ~cc;  dd = #50 ~dd;  end  initial begin  #1000  $finish;  end  endmodule |

1. Design Source Code:

inv 모듈은 4개의 입력(a, b, c, d)과 3개의 출력(e, f, g)을 가지고 있다. e는 a와 b의 OR 연산의 NOT 결과이다. f는 e와 c의 OR 연산의 NOT 결과이다. g는 f와 d의 OR 연산의 NOT 결과이다.

2. Simulation Source Code:

위의 NAND 연산과 동일한 코드이다. inv\_tb 모듈은 inv 모듈을 테스트하기 위한 테스트벤치(Testbench) 역할을 한다. 먼저 inv 모듈을 인스턴스화하고, 입력 및 출력 신호를 연결한다. 다음으로 초기 블록(initial)에서 입력 신호 aa, bb, cc, dd를 초기화한다. always 블록에서는 입력 신호 aa, bb, cc, dd를 일정한 주기로 반전시킨다. 마지막 initial 블록에서는 1000ns(나노초) 후에 시뮬레이션을 종료하는 $finish 명령을 실행한다.

1. Simulation 결과

스크린샷, 텍스트, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드를 시뮬레이션 돌린 결과이다. 시간에 따라 입력 A, B, C, D의 값을 바꾸면 그에 따라 출력 E, F, G의 값이 변화하는 것을 확인할 수 있다. 출력 E는 A와 B가 둘다 0인 경우에만 1의 값을 가진다. A와 B 둘 중 적어도 하나가 1의 값을 가지면 출력 값으로 0을 내보낸다. 출력 F는 E와 C의 값이 둘다 0인 경우에만 1의 값을 가진다. E와 C 둘 중 적어도 하나가 1의 값을 가지면 출력 값으로 0을 내보낸다. 출력 G는 F와 D의 값이 둘다 0인 경우에만 1의 값을 가진다. F와 D 둘 중 적어도 하나가 1의 값을 가지면 출력 값으로 0을 내보낸다. 이 시뮬레이션 결과를 통해 5)의 진리표를 작성할 수 있다. 작성된 진리표를 통해 다중입력 NOR 게이트의 최종 출력 G는 입력 값 A, B, C, D가 (0000), (0010), (0110), (1010), (1110)인 경우에 1의 값을 가지고, 나머지 경우에는 0의 값을 가진다는 것을 알 수 있다. 이를 통해 이전 주차에서 실습했던 OR 게이트는 다단으로 설계해도 모든 입력이 0이면 0을 출력한다는 OR 게이트의 특성을 유지하지만, NOR 게이트는 다단으로 설계하는 경우 모든 입력이 0인 경우에만 1을 출력한다는 NOR 게이트의 특성을 유지하지 않음을 확인할 수 있었다.

1. Schematic Diagram

도표, 라인이(가) 표시된 사진

자동 생성된 설명

위는 Vivado에서 생성한 4-input NOR gate의 Schematic Diagram이다. 입력 A와 B를 NOR(OR+NOT) gate에 연결해 E를 출력하고, 입력 E와 C를 NOR(OR+NOT) gate에 연결해 F를 출력하고, 입력 F와 D를 NOR(OR+NOT) gate에 연결해 G를 출력한다는 점에서 작성한 코드가 (B)와 동일하게 동작하는 것을 알 수 있다.

1. 4-input NOR gate의 진리표

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

**4.**

**4-input XOR gate**

도표, 라인, 스케치, 기술 도면이(가) 표시된 사진

자동 생성된 설명

1. Boolean식

4-input XOR gate의 경우 A와 B를 XOR gate에 연결해 E를 출력하고, E와 C를 XOR gate에 연결해 F를 출력하고, F와 D를 XOR gate에 연결해 G를 출력한다. 따라서 (B)의 Boolean식은, , (, , )의 순서로 구성할 수 있다.

1. Verilog 코딩

위의 Boolean식에 따라 Verilog 코드를 작성한 결과는 다음과 같다.

|  |  |
| --- | --- |
| Design Source Code | Simulation Source Code |
| `timescale 1ns / 1ps  module 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 begin  aa = 1'b0;  bb = 1'b0;  cc = 1'b0;  dd = 1'b0;  end  always begin  aa = #20 ~aa;  bb = #30 ~bb;  cc = #40 ~cc;  dd = #50 ~dd;  end  initial begin  #1000  $finish;  end  endmodule |

1. Design Source Code:

inv 모듈은 4개의 입력(a, b, c, d)과 3개의 출력(e, f, g)을 가지고 있다. e는 a와 b의 XOR 연산의 결과이다. f는 e와 c의 XOR 연산의 결과이다. g는 f와 d의 XOR 연산의 결과이다.

2. Simulation Source Code:

위의 NAND, NOR 연산과 동일한 코드이다. inv\_tb 모듈은 inv 모듈을 테스트하기 위한 테스트벤치(Testbench) 역할을 한다. 먼저 inv 모듈을 인스턴스화하고, 입력 및 출력 신호를 연결한다. 다음으로 초기 블록(initial)에서 입력 신호 aa, bb, cc, dd를 초기화한다. always 블록에서는 입력 신호 aa, bb, cc, dd를 일정한 주기로 반전시킨다. 마지막 initial 블록에서는 1000ns(나노초) 후에 시뮬레이션을 종료하는 $finish 명령을 실행한다.

1. Simulation 결과

스크린샷, 텍스트, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드를 시뮬레이션 돌린 결과이다. 시간에 따라 입력 A, B, C, D의 값을 바꾸면 그에 따라 출력 E, F, G의 값이 변화하는 것을 확인할 수 있다. 출력 E는 A와 B의 값이 서로 다른 경우에만 1의 값을 가진다. A와 B의 값이 동일하면 출력 값으로 0을 내보낸다. 출력 F도 마찬가지로 E와 C의 값이 서로 다른 경우에만 1의 값을 가진다. E와 C의 값이 동일하면 출력 값으로 0을 내보낸다. 출력 G도 F와 D의 값이 서로 다른 경우에만 1의 값을 가진다. F와 D의 값이 동일하면 출력 값으로 0을 내보낸다. 이 시뮬레이션 결과를 통해 5)의 진리표를 작성할 수 있다. 작성된 진리표를 통해 다중입력 XOR 게이트의 최종 출력 G는 입력 값 A, B, C, D가 (0001), (0010), (0100), (1000), (0111), (1011), (1101), (1110)인 경우에 1의 값을 가지고, 나머지 경우에는 0의 값을 가진다는 것을 알 수 있다. 즉 다중입력 XOR 게이트의 입력 값 중 1의 개수가 홀수 개이면 출력으로 1의 값을 가지고 1의 개수가 짝수 개이면 출력으로 0의 값을 가지는 특성을 확인할 수 있다.

1. Schematic Diagram

도표, 라인, 평면도, 그래프이(가) 표시된 사진

자동 생성된 설명

위는 Vivado에서 생성한 4-input XOR gate의 Schematic Diagram이다. 입력 A와 B를 XOR gate에 연결해 E를 출력하고, 입력 E와 C를 XOR gate에 연결해 F를 출력하고, 입력 F와 D를 XOR gate에 연결해 G를 출력한다는 점에서 작성한 코드가 (B)와 동일하게 동작하는 것을 알 수 있다.

1. 4-input XOR gate의 진리표

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

**5.**

**4-input AOI gate**

도표, 스케치, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명

1. Boolean식

4-input AOI gate의 경우 A와 B를 AND gate에 연결해 E를 출력하고, C와 D를 AND gate에 연결해 F를 출력하고, E와 F를 NOR gate에 연결해 G를 출력한다. 따라서 (B)의 Boolean식은 , , 의 순서로 구성할 수 있다.

1. Verilog 코딩

위의 Boolean식에 따라 Verilog 코드를 작성한 결과는 다음과 같다.

|  |  |
| --- | --- |
| Design Source Code | Simulation Source Code |
| `timescale 1ns / 1ps  module inv(  input a, b, c, d,  output e, f, g  );  assign e=a&b;  assign f=c&d;  assign g=~(e|f);  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 begin  aa = 1'b0;  bb = 1'b0;  cc = 1'b0;  dd = 1'b0;  end  always begin  aa = #20 ~aa;  bb = #30 ~bb;  cc = #40 ~cc;  dd = #50 ~dd;  end  initial begin  #1000  $finish;  end  endmodule |

1. Design Source Code:

inv 모듈은 4개의 입력(a, b, c, d)과 3개의 출력(e, f, g)을 가지고 있다. e는 a와 b의 AND 연산의 결과이다. f는 c와 d의 AND 연산의 결과이다. g는 e와 f의 OR 연산의 NOT 결과이다.

2. Simulation Source Code:

위의 NAND, NOR, XOR 연산과 동일한 코드이다. inv\_tb 모듈은 inv 모듈을 테스트하기 위한 테스트벤치(Testbench) 역할을 한다. 먼저 inv 모듈을 인스턴스화하고, 입력 및 출력 신호를 연결한다. 다음으로 초기 블록(initial)에서 입력 신호 aa, bb, cc, dd를 초기화한다. always 블록에서는 입력 신호 aa, bb, cc, dd를 일정한 주기로 반전시킨다. 마지막 initial 블록에서는 1000ns(나노초) 후에 시뮬레이션을 종료하는 $finish 명령을 실행한다.

1. Simulation 결과

스크린샷, 텍스트, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드를 시뮬레이션 돌린 결과이다. 시간에 따라 입력 A, B, C, D의 값을 바꾸면 그에 따라 출력 E, F, G의 값이 변화하는 것을 확인할 수 있다. 출력 E는 A와 B가 둘다 1인 경우에만 1의 값을 가진다. 출력 F는 C와 D가 둘다 1인 경우에만 1의 값을 가진다. 출력 G는 E와 F가 모두 0인 경우 1의 값을 가지고, E와 F 둘 중 적어도 하나가 1의 값을 가지면 출력 값으로 0을 내보낸다. 즉 출력 G가 1의 값을 가지려면 입력 A, B 둘 중 적어도 하나가 0이고, 입력 C, D 둘 중 적어도 하나가 0이어야 한다. 앞의 두 조건 중 하나라도 만족하지 않는 경우에는 출력 G가 0의 값을 가지게 된다. 시뮬레이션 결과를 통해 5)의 진리표를 작성할 수 있다. 작성된 진리표를 통해 다중입력 AOI 게이트의 최종 출력 G는 입력 값 A, B, C, D가 (0011), (0111), (1011), (1100), (1101), (1110), (1111)인 경우 처럼 앞의 두 조건 중 하나라도 만족하지 않는 경우 0의 값을 가지고, 나머지 경우 두 조건 모두 만족하기에 1의 값을 가진다는 것을 알 수 있다.

1. Schematic Diagram

도표, 텍스트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

위는 Vivado에서 생성한 4-input AOI gate의 Schematic Diagram이다. 입력 A와 B를 AND gate에 연결해 E를 출력하고, 입력 C와 D를 AND gate에 연결해 F를 출력하고, 입력 E와 F를 NOR(OR+NOT) gate에 연결해 G를 출력한다는 점에서 작성한 코드가 (B)와 동일하게 동작하는 것을 알 수 있다.

1. 4-input AOI gate의 진리표

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

**6.**

3주차에서 실습했던 AND 게이트는 다단으로 설계해도 입력 개수에 상관없이 모든 입력이 1이면 1을 출력한다는 AND 게이트의 특성을 유지하지만, 이번 주차에 실습한 NAND 게이트는 다단으로 설계하는 경우 모든 입력이 1인 경우에만 0을 출력한다는 NAND 게이트의 특성을 유지하지 않음을 실험적으로 확인할 수 있었다. 이는 NAND 게이트를 다단으로 설계할 경우 Boolean 식이 G = (ABCD)’이 아닌 G = (((AB)’C)’D)’로 구성되기 때문이다.

마찬가지로 3주차에서 실습했던 OR 게이트는 다단으로 설계해도 입력 개수에 상관 없이 모든 입력이 0이면 0을 출력한다는 OR 게이트의 특성을 유지하지만, 이번 주차에 실습한 NOR 게이트는 다단으로 설계하는 경우 모든 입력이 0인 경우에만 1을 출력한다는 NOR 게이트의 특성을 유지하지 않음을 실험적으로 확인할 수 있었다. 이는 NOR 게이트를 다단으로 설계할 경우 Boolean 식이 G = (A+B+C+D)’이 아닌 G = (((A+B)’+C)’+D)’로 구성되기 때문이다.

XOR 게이트의 경우는 AND, OR 게이트와 마찬가지로 다단으로 설계해도 그 특성을 유지한다는 것을 확인할 수 있었다. 다중입력 XOR 게이트는 입력 값 중 1의 개수가 홀수 개이면 출력으로 1의 값을 가지고 1의 개수가 짝수 개이면 출력으로 0의 값을 가지는 특성을 가진다. 이는 Boolean 식이 G = A⊕B⊕C⊕D로 구성되든 G = (((A⊕B)⊕C)⊕D)로 구성되든 결합 법칙에 의해 동일한 결과 값을 내놓기 때문이다.

**7.**

XOR 게이트의 경우 실제 boolean식은 , , 로 나타나지만 코드 작성 시에는 XOR 연산자인 ^를 사용해 (case 1) E = A^B, F = E^C, G = F^D와 같이 작성했다. 그대신 boolean식과 동일하게 (case 2) E = (A&(~B))|((~A)&B), F=(E&(~C))|((~E)&C), G=(F&(~D))|((~F)&D) 꼴로 코드를 작성해도 동일한 시뮬레이션 결과를 얻을 수 있다는 것을 확인했다.

case 1

스크린샷이(가) 표시된 사진

자동 생성된 설명

case 2

스크린샷이(가) 표시된 사진

자동 생성된 설명

또한 다중입력 XNOR 게이트는 다단으로 설계하는 경우 그 특성을 유지하는지 알아보았다.

1. Boolean식

4-input XNOR gate의 경우 A와 B를 XNOR gate에 연결해 E를 출력하고, E와 C를 XNOR gate에 연결해 F를 출력하고, F와 D를 XNOR gate에 연결해 G를 출력한다. 따라서 (B)의 Boolean식은 , , 의 순서로 구성할 수 있다.

1. Verilog 코딩

위의 Boolean식에 따라 Verilog 코드를 작성한 결과는 다음과 같다.

|  |  |
| --- | --- |
| Design Source Code | Simulation Source Code |
| `timescale 1ns / 1ps  module 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 begin  aa = 1'b0;  bb = 1'b0;  cc = 1'b0;  dd = 1'b0;  end  always begin  aa = #20 ~aa;  bb = #30 ~bb;  cc = #40 ~cc;  dd = #50 ~dd;  end  initial begin  #1000  $finish;  end  endmodule |

1. Simulation 결과

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명 위에서 작성한 Verilog 코드를 시뮬레이션 돌린 결과이다. 시간에 따라 입력 A, B, C, D의 값을 바꾸면 그에 따라 출력 E, F, G의 값이 변화한다. 이를 통해 XNOR 게이트도 AND, OR, XOR 게이트와 마찬가지로 다단으로 설계해도 그 특성을 유지한다는 것을 확인할 수 있었다. 다중입력 XNOR 게이트는 XOR 게이트와 반대로 입력 값 중 1의 개수가 짝수 개이면 출력으로 1의 값을 가지고 1의 개수가 홀수 개이면 출력으로 0의 값을 가지는 특성을 가진다.