**4주차 결과 보고서**

남민혁

**1. 실험 목적**

4주차 실험의 목적은 아래와 같다.

* NAND/NOR/XOR Gate의 동작의 이해 및 확인
* Verilog를 사용하여 다중입력 NAND/NOR Gate 및 XOR Gate 구현
* 입력 신호 생성 후 Simulation을 통하여 구현된 각 Gate 동작 확인

**2. 4-input NAND gate의 simulation 결과 및 과정에 대해서 설명하시오. (4 input, 3 output)[4장 ppt 6,7 page 참조 , 진리표 작성]**

**-Boolean 식**

|  |  |
| --- | --- |
| (A) | (B) |
|  |  |

**-Verilog 코딩**

|  |  |  |
| --- | --- | --- |
|  | (A) | (B) |
| Verilog | `timescale 1ns / 1ps  module four\_input\_nand\_gate\_a(  input a,  input b,  input c,  input d,  output e  );  assign e = ~(a & b & c & d);  endmodule | `timescale 1ns / 1ps  module four\_input\_nand\_gate\_b(  input a,  input b,  input c,  input d,  output e,  output f,  output g  );  assign e = ~(a & b);  assign f = ~(c & e);  assign g = ~(d & f);  endmodule |
| Test Bench | `timescale 1ns / 1ps  module four\_input\_nand\_gate\_a\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee;  four\_input\_nand\_gate\_a u\_four\_input\_nand\_gate\_a(  .a(aa),  .b(bb),  .c(cc),  .d(dd),  .e(ee));  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  always aa = #400 ~aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  initial begin  #800  $finish;  end  endmodule | `timescale 1ns / 1ps  module four\_input\_nand\_gate\_b\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee;  wire ff;  wire gg;  four\_input\_nand\_gate\_b u\_four\_input\_nand\_gate\_b(  .a(aa),  .b(bb),  .c(cc),  .d(dd),  .e(ee),  .f(ff),  .g(gg));  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  always aa = #400 ~aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  initial begin  #800  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

|  |  |
| --- | --- |
| A | Graphical user interface  Description automatically generated |
| B | Graphical user interface  Description automatically generated |

**-진리표(A)**

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

**-진리표(B)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input A | Input B | Input C | Input D | Output E | Output F | Output 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 gate의 simulation 결과 및 과정에 대해서 설명하시오. (4 input, 3 output)[4장 ppt 8,9 page 참조 , 진리표 작성]**

**-Boolean 식**

|  |  |
| --- | --- |
| (A) | (B) |
|  |  |

**-Verilog 코딩**

|  |  |  |
| --- | --- | --- |
|  | (A) | (B) |
| Verilog | `timescale 1ns / 1ps  module four\_input\_nor\_gate\_a(  input a,  input b,  input c,  input d,  output e  );  assign e = ~(a | b | c | d);  endmodule | `timescale 1ns / 1ps  module four\_input\_nor\_gate\_b(  input a,  input b,  input c,  input d,  output e,  output f,  output g  );  assign e = ~(a | b);  assign f = ~(c | e);  assign g = ~(d | f);  endmodule |
| Test Bench | `timescale 1ns / 1ps  module four\_input\_nor\_gate\_a\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee;  four\_input\_nor\_gate\_a u\_four\_input\_nor\_gate\_a(  .a(aa),  .b(bb),  .c(cc),  .d(dd),  .e(ee));  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  always aa = #400 ~aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  initial begin  #800  $finish;  end  endmodule | `timescale 1ns / 1ps  module four\_input\_nor\_gate\_b\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee;  wire ff;  wire gg;  four\_input\_nor\_gate\_b u\_four\_input\_nor\_gate\_b(  .a(aa),  .b(bb),  .c(cc),  .d(dd),  .e(ee),  .f(ff),  .g(gg));  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  always aa = #400 ~aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  initial begin  #800  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

|  |  |
| --- | --- |
| A | Graphical user interface  Description automatically generated |
| B | Graphical user interface  Description automatically generated |

**-진리표 (A)**

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

**-진리표 (B)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input A | Input B | Input C | Input D | Output E | Output F | Output 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 gate의 simulation 결과 및 과정에 대해서 설명하시오. (4 input, 3 output)[3장 ppt 10,11 page 참조 , 진리표 작성]**

**-Boolean 식**

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

**-Verilog 코딩**

|  |  |  |
| --- | --- | --- |
|  | (A) | (B) |
| Verilog | `timescale 1ns / 1ps  module four\_input\_xor\_gate\_a(  input a,  input b,  input c,  input d,  output e  );  assign e = a ^ b ^ c ^ d;  endmodule | `timescale 1ns / 1ps  module four\_input\_xor\_gate\_b(  input a,  input b,  input c,  input d,  output e,  output f,  output g  );  assign e = a ^ b;  assign f = c ^ e;  assign g = d ^ f;  endmodule |
| Test Bench | `timescale 1ns / 1ps  module four\_input\_xor\_gate\_a\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee;  four\_input\_xor\_gate\_a u\_four\_input\_xor\_gate\_a(  .a(aa),  .b(bb),  .c(cc),  .d(dd),  .e(ee));  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  always aa = #400 ~aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  initial begin  #800  $finish;  end  endmodule | `timescale 1ns / 1ps  module four\_input\_xor\_gate\_b\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee;  wire ff;  wire gg;  four\_input\_xor\_gate\_b u\_four\_input\_xor\_gate\_b(  .a(aa),  .b(bb),  .c(cc),  .d(dd),  .e(ee),  .f(ff),  .g(gg));  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  always aa = #400 ~aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  initial begin  #800  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

|  |  |
| --- | --- |
| A | Graphical user interface  Description automatically generated |
| B | Graphical user interface  Description automatically generated |

**-진리표 (A)**

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

**-진리표 (B)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 | 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 gate의 simulation 결과 및 과정에 대해서 설명하시오. (4 input, 3 output)[4장 ppt 12,13 page 참조, 진리표 작성]**

**-Boolean 식**

**-Verilog 코딩**

|  |  |
| --- | --- |
|  | (A) |
| Verilog | `timescale 1ns / 1ps  module four\_input\_aoi\_gate(  input a,  input b,  input c,  input d,  output e,  output f,  output g  );  assign e = a & b;  assign f = c & d;  assign g = ~(e | f);  endmodule |
| Test Bench | `timescale 1ns / 1ps  module four\_input\_aoi\_gate\_tb;  reg aa;  reg bb;  reg cc;  reg dd;  wire ee;  wire ff;  wire gg;  four\_input\_aoi\_gate u\_four\_input\_aoi\_gate(  .a(aa),  .b(bb),  .c(cc),  .d(dd),  .e(ee),  .f(ff),  .g(gg));  initial aa = 1'b0;  initial bb = 1'b0;  initial cc = 1'b0;  initial dd = 1'b0;  always aa = #400 ~aa;  always bb = #200 ~bb;  always cc = #100 ~cc;  always dd = #50 ~dd;  initial begin  #800  $finish;  end  endmodule |

**-Simulation 출력 결과 비교**

Graphical user interface

Description automatically generated

**-진리표**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input A | Input B | Input C | Input D | Output E | Output F | Output 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. 결과 검토 및 논의사항**

지난 실험에서 구현한 AND/OR 게이트와 달리 이번 실험에서 구현한 NAND/NOR/XOR 게이트는 한번에 모든 입력을 게이트에 넣은 경우와 순차적으로 여러 단계를 거쳐서 넣는 것의 결과가 다르게 나타났다. 이에 따라 시뮬레이션의 결과가 다르게 나타나는 것을 확인할 수 있었다.

**7. 추가 이론 조사 및 작성**

본 실험에서 구현한 NAND/NOR 게이트는 모든 논리게이트를 표현할 수 있는 universal 게이트로서 다양한 논리회로에 사용된다. 지난 시간에 만든 AND와 OR 게이트 역시 NAND와 NOR로 표현 될 수 있으며, 이는 아래와 같다.

-AND게이트

A picture containing application

Description automatically generated

-OR 게이트

A picture containing diagram

Description automatically generated

**8. 참고 문헌**

강석태, “Verilog HDL Summary”, http://vlsi.hongik.ac.kr/lecture/%EC%8B%A4%ED%97%98/Verilog\_Summary.pdf.

한국기술교육대학교 전기전자통신공학부, “디지털시스템 설계 및 실습” Verilog HDL 문법 강의자료, https://cms3.koreatech.ac.kr/sites/yjjang/down/dsys11/M01\_VerilogHDL01.pdf.

장영조, “디지털공학 및 실습 6장 강의자료: 부울함수의 간소화”, https://cms3.koreatech.ac.kr/sites/yjjang/down/digi09/ch06.pdf.