3주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231609 이름: 정희선

**1.**

**FPGA의** **동작법**을 알아보자.

1. **Verilog coding**: Verilog와 같은 HDL로 회로를 설계한다. 이 단계에서는 디지털 회로의 동작을 정의한다.
2. **Run synthesis**: Verilog 코드를 gate-level 회로로 변환한다. 이 단계에서 이전에 작성한 코드를 하드웨어에 매핑한다.
3. **Device/Pin assignment**: 이전 단계에서 설계한 각 신호를 어떤 핀에 연결할지 결정한다.
4. **Synthesis/Implement:** Place and route 작업이 이루어진다. 이 단계에서 논리회로를 FPGA 내부의 논리 블록에 배치한다. 또 논리 블록들 간의 신호 경로를 설정한다.
5. **Device configuration:** FPGA에 회로를 업로드하고, 회로대로 동작하도록 프로그래밍한다.

**2.**

3-input AND gate

**1) Boolean식:** *D = A⋅B, E = D⋅C*

**2) Verilog 코딩**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**3) Simulation 결과**

스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**4) FPGA 동작**

Synthesis 단계를 통해 FPGA에 코드를 올려 게이트 레벨로 구현한다.

**5) 진리표**

모든 입력이 1일 때만 1을 출력한다.

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

**3.**

4-input AND gate

**1) Boolean식:** *E = A⋅B, F = E⋅C, G = F⋅D*

**2) Verilog 코딩**

*`timescale 1ns / 1ps*

*module FourAnd(*

*input a,b,c,d,*

*output e,f,g*

*);*

*assign e=a&b;*

*assign f=e&c;*

*assign g=f&d;*

*endmodule*

**3) Simulation 결과**

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

자동 생성된 설명**

*`timescale 1ns / 1ps*

*module FourAnd\_tb(*

*);*

*reg aa,bb,cc,dd;*

*wire ee,ff,gg;*

*FourAnd u\_test(*

*.a(aa),*

*.b(bb),*

*.c(cc),*

*.d(dd),*

*.e(ee),*

*.f(ff),*

*.g(gg)*

*);*

*initial begin*

*aa=1'b0;*

*bb=1'b0;*

*cc=1'b0;*

*dd=1'b0;*

*end*

*always@(aa or bb or cc or dd) begin*

*aa <= #50 ~aa;*

*bb <= #100 ~bb;*

*cc <= #150 ~cc;*

*dd <= #200 ~dd;*

*end*

*initial begin*

*#1000*

*$finish;*

*end*

*endmodule*

**4) FPGA 동작**

Synthesis 단계를 통해 FPGA에 코드를 올려 게이트 레벨로 구현한다.

**5) 진리표**

진리표는 위와 유사한데, 모든 입력이 1일 때만 1을 출력한다.

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

**4.**

3-input OR gate

**1) Boolean식:** *D = A+B, E = D+C*

**2) Verilog 코딩**

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

**3) Simulation 결과**

스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**4) FPGA 동작**

Synthesis 단계를 통해 FPGA에 코드를 올려 게이트 레벨로 구현한다.

**5) 진리표**

진리표에서는 입력 값 중 1이 존재하면 출력이 1이 된다.

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

**5.**

4-input OR gate

**1) Boolean식:** *E = A+B, F = E+C, G = F+D*

**2) Verilog 코딩**

*`timescale 1ns / 1ps*

*module FourOr(*

*input a,b,c,d,*

*output e,f,g*

*);*

*assign e=a||b;*

*assign f=e||c;*

*assign g=f||d;*

*endmodule*

**3) Simulation 결과**

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

자동 생성된 설명**

*`timescale 1ns / 1ps*

*module FourOr\_tb(*

*);*

*reg aa,bb,cc,dd;*

*wire ee,ff,gg;*

*FourOr u\_test(*

*.a(aa),*

*.b(bb),*

*.c(cc),*

*.d(dd),*

*.e(ee),*

*.f(ff),*

*.g(gg)*

*);*

*initial begin*

*aa=1'b0;*

*bb=1'b0;*

*cc=1'b0;*

*dd=1'b0;*

*end*

*always@(aa or bb or cc or dd) begin*

*aa <= #50 ~aa;*

*bb <= #100 ~bb;*

*cc <= #150 ~cc;*

*dd <= #200 ~dd;*

*end*

*initial begin*

*#1000*

*$finish;*

*end*

*endmodule*

**4) FPGA 동작**

Synthesis 단계를 통해 FPGA에 코드를 올려 게이트 레벨로 구현한다.

**5) 진리표**

진리표는 위와 유사한데, 모든 입력 값 중 1이 존재하면 결과는 1이 된다.

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

**6.**

* And gate는 입력 중 하나라도 0이면 출력이 0이 나온다는 것을 알 수 있었다. 즉, 모든 입력이 1일 때만 출력이 1이 되었다.
* Or gate는 입력 중 하나라도 1이면 출력이 1이되는 것을 알 수 있었다.
* 게이트 수가 늘어날수록 특정 출력의 경우가 적어진다.
* And gate는 입력이 많아질수록 출력이 1이되는 경우가 줄어들었다.
* OR gate는 입력이 많아질수록 출력이 0이되는 경우가 줄어들었다.

**7.**

- 지연시간: FPGA에서 논리 게이트 작동 시, 입력에서 출력까지 신호가 전달되는 시간에 지연이 발생한다. 이 때 게이트의 개수에 따라 지연 시간이 누적될 수 있다.

- 경합 조건: FPGA 내부에서 여러 입력 신호가 동시에 변화하면, 특정 신호는 타이밍 차이로 잘못된 출력을 유발하는 경우가 있다. 이를 방지하기 위해 신호의 정확한 타이밍을 제어하는 것이 필요하다.