11주차 결과보고서

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

**1.**

**1) RS Flip-Flop (NAND)**

RS 플립-플롭은 가장 기본적인 형태의 플립 플롭으로, 클럭(CLK)과 두 개의 입력(R과 S), 두 개의 출력(Q와 Q')을 가지고 있다. Clock 신호가 1이 되는 경우에만 다음과 같이 동작해 상태 값을 업데이트한다. R이 1이고 S가 0이면 Q 값을 0으로 reset 하고, R이 0이고 S가 1이면 값을 1로 set 하며, 둘다 0이면 기존 값을 그대로 유지한다. 둘다 1의 값을 가지는 경우는 플립 플롭의 동작이 정의될 수 없으므로 허용하지 않는다.

* Charateristic Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | |  | | **Output** | | **State of the output** |
| **CLK** | **R** | **S** | **Qn** | **Q’n** | **Qn+1** | **Q’n+1** |
| 0 | ― | ― | 1 | 1 | Qn | Q’n | Hold no change  previous state Reset |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | Reset |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | Set |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | Invalid Forbidden |

위에서 설명한 RS 플립 플롭의 특성을 표로 나타낸 것이다.

* Verilog source

|  |  |
| --- | --- |
| Design Source Code | Simulation Source Code |
| `timescale 1ns / 1ps  module rsff1(  input clk, r, s,  output q, \_q  );  wire out1;  wire out2;  assign out1 = ~(r&clk);  assign out2 = ~(s&clk);  assign q = ~(out2&\_q);  assign \_q = ~(out1&q);  endmodule | `timescale 1ns / 1ps  module rsff1\_tb;  reg clk, r, s;  wire q, \_q;  rsff1 u\_rsff1 (  .clk (clk ),  .r (r ),  .s (s ),  .q (q ),  .\_q (\_q )  );  initial begin  clk = 1'b0;  r = 1'b0;  s = 1'b0;  end  always clk = #50 ~clk;  always r = #200 ~r;  always s = #100 ~s;  initial begin  #1000  $finish;  end  endmodule |

위의 특성표를 바탕으로 NAND 게이트를 사용하는 RS 플립 플롭의 Verilog 코드를 작성한 결과이다. rsff1 모듈은 3개의 입력(clk, r, s)과 2개의 출력(q, \_q)을 가지고 있다. 이때 clk는 Clock 신호를 의미하고, r은 Reset, s는 Set 입력을 나타낸다. 또한 중간 결과를 저장하기 위한 두 개의 wire 변수(out1, out2)가 사용된다. out1은 r과 clk의 NAND 연산 결과이고, out2는 s와 clk의 NAND 연산 결과이다. 또한 q는 out2와 \_q의 NAND 연산 결과이며, \_q는 out1과 q의 NAND 연산 결과이다.

* Schematic Diagram

도표, 라인, 평면도, 폰트이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드로 Schematic Diagram을 생성한 결과이다. 이를 통해 RS 플립 플롭이 총 4개의 NAND 게이트로 구성된 것을 확인할 수 있다.

* Simulation 결과

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드를 시뮬레이션 돌린 결과이다. 시간에 따라 입력 clk, r, s의 값을 바꾸면 그에 따라 출력 q, \_q의 값이 변화하는 것을 확인할 수 있다. 구현한 RS 플립 플롭이 각 입력 값에 따라 어떻게 동작하는지 좀 더 자세히 살펴보겠다.

* 출력 예시

스크린샷이(가) 표시된 사진

자동 생성된 설명

> 먼저 clk, r, s가 순서대로 0, 0, 0인 경우 출력의 초기 값을 정해놓지 않아 q, \_q가 X로 표시된다.

스크린샷, 사각형, 다채로움이(가) 표시된 사진

자동 생성된 설명

> clk이 1로 바뀌고 r, s가 0, 0인 경우 마찬가지로 출력의 초기 값이 없어 q, \_q가 X로 표시된다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

> clk이 0의 값을 가지기에 상태 업데이트가 되지 않아 r이 0, s가 1이 되어도 출력 값엔 변화가 없다. 따라서 q, \_q는 이전과 마찬가지로 X로 표시된다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 1로 바뀌고 이때 r은 0, s는 1의 값을 가지므로 출력 q가 1로 set 된다. 출력 \_q는 그 반대인 0이 된다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

> clk이 0이 되어 이전 출력 값을 유지한다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 1이 되고 이때 r은 1, s는 0의 값을 가진다. 따라서 출력 q는 0으로 reset 되고, 출력 \_q는 1이 된다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 0이 되어 이전 출력 값을 유지한다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 1이고 r과 s 둘다 1의 값을 가지는 경우이다. 이는 RS 플립 플롭에서 허용하지 않는 입력 값으로 이 경우는 고려하지 않는다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 0이 되어 이전 출력 값을 유지해야 하지만, 이전 입력 값이 1, 1, 1이기 때문에 이 경우 또한 고려하지 않는다.

스크린샷, 사각형, 다채로움이(가) 표시된 사진

자동 생성된 설명

> clk이 1, r이 0, s가 0의 값을 가져 이전 출력 값을 유지한다.

* Truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| **입력순서** | **R** | **S** | **Q** | **~Q** |
| (1) | 0 | 1 | 1 | 0 |
| (2) | 0 | 0 | 1 | 0 |
| (3) | 1 | 0 | 0 | 1 |
| (4) | 0 | 0 | 0 | 1 |
| (5) | 1 | 0 | 0 | 1 |
| (6) | 1 | 1 | ― | ― |

시뮬레이션 결과를 바탕으로 강의 자료에 있는 truth table의 빈칸을 완성한 결과이다. 이는 입력 순서에 따른 RS 플립 플롭의 동작을 보여주며, 특성표와 비교했을 때 설계한 플립 플롭이 의도한 대로 올바르게 작동하고 있음을 확인할 수 있다.

**2) RS Flip-Flop (NOR)**

* Charateristic Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | |  | | **Output** | | **State of the output** |
| **CLK** | **R** | **S** | **Qn** | **Q’n** | **Qn+1** | **Q’n+1** |
| 0 | ― | ― | 1 | 1 | Qn | Q’n | Hold no change  previous state Reset |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | Reset |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | Set |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | Invalid Forbidden |

NOR 게이트를 이용해 RS 플립 플롭을 구현하는 경우에도 NAND 게이트로 구현하는 경우와 동일한 특성표를 가진다.

* Verilog source

|  |  |
| --- | --- |
| Design Source Code | Simulation Source Code |
| `timescale 1ns / 1ps  module rsff2(  input clk, r, s,  output q, \_q  );  wire out1;  wire out2;  assign out1 = r&clk;  assign out2 = s&clk;  assign q = ~(out1|\_q);  assign \_q = ~(out2|q);  endmodule | `timescale 1ns / 1ps  module rsff2\_tb;  reg clk, r, s;  wire q, \_q;  rsff2 u\_rsff2 (  .clk (clk ),  .r (r ),  .s (s ),  .q (q ),  .\_q (\_q )  );  initial begin  clk = 1'b0;  r = 1'b0;  s = 1'b0;  end  always clk = #50 ~clk;  always r = #200 ~r;  always s = #100 ~s;  initial begin  #1000  $finish;  end  endmodule |

위의 특성표를 바탕으로 NOR 게이트를 사용하는 RS 플립 플롭의 Verilog 코드를 작성한 결과이다. rsff2 모듈은 3개의 입력(clk, r, s)과 2개의 출력(q, \_q)을 가지고 있다. 이때 clk는 Clock 신호를 의미하고, r은 Reset, s는 Set 입력을 나타낸다. 또한 중간 결과를 저장하기 위한 두 개의 wire 변수(out1, out2)가 사용된다. out1은 r과 clk의 AND 연산 결과이고, out2는 s와 clk의 AND 연산 결과이다. q는 out1과 \_q의 NOR 연산 결과이며, \_q는 out2와 q의 NOR 연산 결과이다.

* Schematic Diagram

도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드로 Schematic Diagram을 생성한 결과이다. 이를 통해 RS 플립 플롭이 2개의 NOR 게이트와 2개의 AND 게이트로 구성된 것을 확인할 수 있다.

* Simulation 결과

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드를 시뮬레이션 돌린 결과이다. 시간에 따라 입력 clk, r, s의 값을 바꾸면 그에 따라 출력 q, \_q의 값이 변화하는 것을 확인할 수 있다. 구현한 RS 플립 플롭이 각 입력 값에 따라 어떻게 동작하는지 좀 더 자세히 살펴보겠다.

* 출력 예시

스크린샷이(가) 표시된 사진

자동 생성된 설명

> 먼저 clk, r, s가 순서대로 0, 0, 0인 경우 출력의 초기 값을 정해놓지 않아 q, \_q가 X로 표시된다.

스크린샷, 사각형, 다채로움이(가) 표시된 사진

자동 생성된 설명

> clk이 1로 바뀌고 r, s가 0, 0인 경우 마찬가지로 출력의 초기 값이 없어 q, \_q가 X로 표시된다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 0의 값을 가지기에 상태 업데이트가 되지 않아 r이 0, s가 1이 되어도 출력 값엔 변화가 없다. 따라서 q, \_q는 이전과 마찬가지로 X로 표시된다.

스크린샷, 사각형, 다채로움이(가) 표시된 사진

자동 생성된 설명

> clk이 1로 바뀌고 이때 r은 0, s는 1의 값을 가지므로 출력 q가 1로 set 된다. 출력 \_q는 그 반대인 0이 된다.

스크린샷, 사각형, 다채로움이(가) 표시된 사진

자동 생성된 설명

> clk이 0이 되어 이전 출력 값을 유지한다.

스크린샷, 사각형, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

> clk이 1이 되고 이때 r은 1, s는 0의 값을 가진다. 따라서 출력 q는 0으로 reset 되고, 출력 \_q는 1이 된다.

스크린샷, 사각형, 다채로움이(가) 표시된 사진

자동 생성된 설명

> clk이 0이 되어 이전 출력 값을 유지한다.

스크린샷, 사각형, 다채로움이(가) 표시된 사진

자동 생성된 설명

> clk이 1이고 r과 s 둘다 1의 값을 가지는 경우이다. 이는 RS 플립 플롭에서 허용하지 않는 입력 값으로 이 경우는 고려하지 않는다.

스크린샷, 사각형, 다채로움이(가) 표시된 사진

자동 생성된 설명

> clk이 0이 되어 이전 출력 값을 유지해야 하지만, 이전 입력 값이 1, 1, 1이기 때문에 이 경우 또한 고려하지 않는다.

스크린샷, 멀티미디어 소프트웨어, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 1, r이 0, s가 0의 값을 가져 이전 출력 값을 유지한다.

* Truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Output** | |
| **입력순서** | **R** | **S** | **Q** | **~Q** |
| (1) | 0 | 1 | 1 | 0 |
| (2) | 0 | 0 | 1 | 0 |
| (3) | 1 | 0 | 0 | 1 |
| (4) | 0 | 0 | 0 | 1 |
| (5) | 1 | 0 | 0 | 1 |
| (6) | 1 | 1 | ― | ― |

시뮬레이션 결과를 바탕으로 강의 자료에 있는 truth table의 빈칸을 완성한 결과이다. 이는 입력 순서에 따른 RS 플립 플롭의 동작을 보여주며, 특성표와 비교했을 때 설계한 플립 플롭이 의도한 대로 올바르게 작동하고 있음을 확인할 수 있다.

**2.**

**D Flip-Flop**

D 플립-플롭은 디지털 논리 회로에서 사용되는 또 다른 종류의 플립 플롭으로, 클럭(CLK)과 하나의입력(D), 두 개의 출력(Q와 Q')을 가지고 있다. Clock 신호가 1이 되는 경우에만 다음과 같이 동작해 상태 값을 업데이트한다. 이는 D가 1이면 출력 Q도 1이 되고 출력 ~Q는 0이 된다. 반대로 D가 0이면 출력 Q도 0이 되고 출력 ~Q는 1이 된다.

* Charateristic Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
| **CLK** | **D** | **Q** | **~Q** |
| 0 | 0 | No Change | |
| 0 | 1 | 1 | 0 |
| 1 | 0 | No Change | |
| 1 | 1 | 0 | 1 |

위에서 설명한 D 플립 플롭의 특성을 표로 나타낸 것이다.

* Verilog source

|  |  |
| --- | --- |
| Design Source Code | Simulation Source Code |
| `timescale 1ns / 1ps  module dff(  input clk, d,  output q, \_q  );  wire out1;  wire out2;  assign out1 = ~d&clk;  assign out2 = d&clk;  assign q = ~(out1|\_q);  assign \_q = ~(out2|q);  endmodule | `timescale 1ns / 1ps  module dff\_tb;  reg clk, d;  wire q, \_q;  dff u\_dff (  .clk (clk ),  .d (d ),  .q (q ),  .\_q (\_q )  );  initial begin  clk = 1'b0;  d = 1'b0;  end  always clk = #50 ~clk;  always d = #100 ~d;  initial begin  #1000  $finish;  end  endmodule |

위의 특성표를 바탕으로 D 플립 플롭의 Verilog 코드를 작성한 결과이다. dff 모듈은 2개의 입력(clk, d)과 2개의 출력(q, \_q)을 가지고 있다. 이때 clk는 Clock 신호를 의미하고, d는 입력 값을 나타낸다. 또한 중간 결과를 저장하기 위한 두 개의 wire 변수(out1, out2)가 사용된다. out1은 d와 clk의 AND 연산 결과의 부정이고, out2는 d와 clk의 AND 연산 결과이다. q는 out1과 \_q의 NOR 연산 결과이며, \_q는 out2와 q의 NOR 연산 결과이다.

* Schematic Diagram

도표, 라인, 평면도, 폰트이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드로 Schematic Diagram을 생성한 결과이다. 이를 통해 D 플립 플롭이 하나의 NOT 게이트, 두 개의 AND 게이트, 두 개의 NOR 게이트로 구성된 것을 확인할 수 있다. 이는 입력 d와 ~d를 NOR 게이트로 구현한 RS 플립 플롭의 두 입력으로 주는 것과 같다.

* Simulation 결과

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

위에서 작성한 Verilog 코드를 시뮬레이션 돌린 결과이다. 시간에 따라 입력 clk, d의 값을 바꾸면 그에 따라 출력 q, \_q의 값이 변화하는 것을 확인할 수 있다. 구현한 D 플립 플롭이 각 입력 값에 따라 어떻게 동작하는지 좀 더 자세히 살펴보겠다.

* 출력 예시

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> 먼저 clk, d가 0, 0인 경우 출력의 초기 값을 정해놓지 않아 q, \_q가 X로 표시된다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 1이 되고, 이때 d는 0의 값을 가진다. 출력 q는 0으로 업데이트 되고, 출력 \_q는 1이 된다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 0이 되어 이전 출력 값을 유지한다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 1이고 d가 1인 경우, 출력 q는 1로 업데이트 되고, 출력 \_q는 0이 된다.

스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명

> clk이 0이 되어 이전 출력 값을 유지한다.

* Truth table

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
| **입력 순서** | **D** | **Q** | **~Q** |
| (1) | 0 | 0 | 1 |
| (2) | 0 | 0 | 1 |
| (3) | 1 | 1 | 0 |
| (4) | 0 | 0 | 1 |
| (5) | 1 | 1 | 0 |
| (6) | 1 | 1 | 0 |

시뮬레이션 결과를 바탕으로 강의 자료에 있는 truth table의 빈칸을 완성한 결과이다. 이는 입력 순서에 따른 D 플립 플롭의 동작을 보여주며, 특성표와 비교했을 때 설계한 플립 플롭이 의도한 대로 올바르게 작동하고 있음을 확인할 수 있다.

**3.**

각 실험에서 Charateristic Table, 시뮬레이션, Truth Table의 결과가 일관되고, 또한 FPGA 연결 시에도 회로가 의도한대로 동작함을 확인했다. 따라서 RS 플립 플롭(NAND, NOR), D 플립 플롭의 회로가 전부 올바르게 구현되었음을 실험적으로 확인할 수 있었다.

실습 당시에 Generate Bitstream 단계에서 오류가 발생해 constraint 파일에 “set\_property ALLOW\_COMBINATORIAL\_LOOPS TRUE [get\_nets q\_OBUF]”라는 코드를 추가해 이를 해결했다. 원인은 플립 플롭 디자인에서 컴비네이셔널 루프(combinatorial loop)가 발생했기 때문인데, 이는 논리 회로에서 입력과 출력이 서로에게 종속되어 무한히 반복되는 상황을 말한다. 이번 실험과 같이 디자인에서 특정한 회로 경로에 대한 컴비네이셔널 루프가 발생해 오류가 나는 경우, 위와 같은 명령어를 사용해 루프를 허용하는 방법으로 문제를 해결할 수 있음을 알게 되었다.

**4.**

Vivado에서 사용되는 주요 제약 명령어들을 몇 가지 조사했다. 이 명령어들은 디자인 제약을 지정하고, 설정하며, 최적화 도구에 대한 특정 동작을 지시하는 데 사용된다.

* create\_clock: 클럭 신호의 주기 및 시스템의 클럭 영역을 정의한다.

ex) create\_clock -period <value> -waveform <waveform> [get\_ports <clock\_port>]

* set\_input\_delay 및 set\_output\_delay: 입력 또는 출력의 레이턴시 제약을 설정한다.

ex1) set\_input\_delay -clock <clock> -min <value> [get\_ports <input\_port>]

ex2) set\_output\_delay -clock <clock> -max <value> [get\_ports <output\_port>]

* read\_verilog 및 write\_checkpoint: Verilog 파일을 읽거나 디자인 상태를 저장한다.

ex1) read\_verilog <file\_name>

ex2) write\_checkpoint -force <checkpoint\_name>

* set\_max\_delay 및 set\_min\_delay: 특정 신호에 대한 최대 및 최소 딜레이를 설정한다.
* set\_false\_path: 특정 경로를 타이밍 분석에서 제외한다.

ex) set\_false\_path -from [get\_cells <source\_cell>] -to [get\_cells <destination\_cell>]

* set\_property BUFFER\_TYPE 및 set\_property REMOVE\_OFFSET을 사용하여 버퍼 인서션 및 제거에 대한 제약을 지정할 수 있다.
* opt\_design: 디자인 최적화를 수행한다.