13주차 결과보고서

전공: 컴퓨터공학과/생명과학과 학년: 9학기 학번: 20181435 이름: 박다희

**1.**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 회로, 디자인이(가) 표시된 사진

자동 생성된 설명**

**도표, 라인, 평면도, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

위의 그림은 차례대로 4-bit shift register의 test bench, design source, simulation, schematic diagram이다. Bit가 하나씩 이동하는 shift register을 Verilog로 구현하기 위해 먼저 초기 출력 값 out을 0000(out[3]out[2]out[1]out[0])로 설정하고 입력 값인 in이 1일 때 out[3]에 1을, in이 0이면 out[3]에 0을 할당하여 하나씩 out[3]에서 out[2], out[1], out[0]순으로 1 bit씩 이동하게 구현하였다. 이 코드로 잘 작동하는지 simulation 결과를 살펴보면 초기화 결정 변수인 rst가 0이고, in이 1일 때는 out[3]도 1로 들어가고, clk의 상승 엣지에 따라 1000, 1100, 0110, 0011, 0001 순으로 bit가 오른쪽으로 shift되는 것을 볼 수 있다. 또한 in이 0일 때는 out[3]에 0이 들어가는 것도 확인할 수 있다. 이를 토대로 output table을 작성하면 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number  Clock Transitions | In | Out[3] | Out[2] | Out[1] | Out[0] |
| 초기 상태 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 | 1 |
| 5 | 0 | 0 | 0 | 0 | 1 |
| 6 | 1 | 1 | 0 | 0 | 0 |
| 7 | 1 | 1 | 1 | 0 | 0 |

이를 FPGA 동작으로도 확인하니 올바르게 동작함을 볼 수 있었다.

**2.**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 텍스트, 도표, 회로이(가) 표시된 사진

자동 생성된 설명도표, 라인, 텍스트, 평면도이(가) 표시된 사진

자동 생성된 설명**

위의 그림은 차례대로 4-bit ring counter의 test bench, design source, simulation, schematic diagram이다. 4개의 flip-flop이 직렬로 연결되어 clk이 1이 될 때마다 1이 그 다음 flip-flop의 출력 값이 되는 4-bit ring counter을 Verilog로 구현하기 위해 초기 값을 0001(out[3]out[2]out[1]out[0])로 설정하고 rst가 1일 때 clk이 1이 될 때마다 out[0]의 값이 out[3]로, out[3]의 값이 out[2]로, out[2]의 값이 out[1]로, out[1]의 값이 out[0]로 동시에 이동하게 Verilog을 작성했다. 변수 rst가 0이 되면 초기값인 0001로 초기화되게 하였고 이 코드가 잘 작동하는 지 확인하기 위해 simulation 결과를 보면, rst가 0일 때 초기값인 0001이 계속 유지되는 것을 볼 수 있다. 초기화 변수 rst가 1일 때는 clk가 1일 때 상승 엣지 때마다 1000, 0100, 0010, 0001, 1000순으로 값이 변하는 것을 볼 수 있다. 이를 토대로 output table을 작성하면 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clock  transition | output | | | |
| out[3] | out[2] | out[1] | out[0] |
| 초기 상태 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 |
| 8 | 0 | 0 | 0 | 1 |

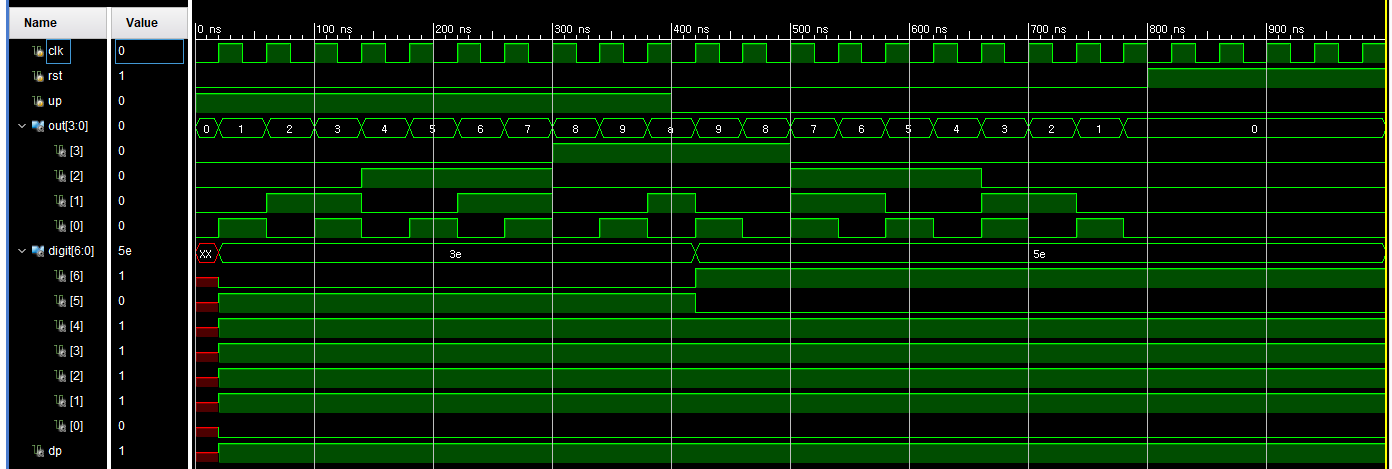
이를 FPGA 동작으로도 확인하니 올바르게 동작함을 볼 수 있었다.

**3.**

**텍스트, 스크린샷, 폰트, 대수학이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

****

**도표, 라인, 평면도, 그래프이(가) 표시된 사진

자동 생성된 설명**

위의 그림은 차례대로 4-bit up/down counter의 test bench, design source, simulation, schematic diagram이다. 4-bit up/down counter을 Verilog로 구현하기 위하여 먼저 초기 값으로 out값을 0000(out[3]out[2]out[1]out[0])로 설정하였고, up counter인지 down counter인지 결정하는 변수인 up을 설정하여 up의 값이 1이면 up counter로 출력인 out의 값에 1을 하나씩 덧셈하였고, up의 값이 0이면 down counter로 출력인 out의 값에 1씩 뺄셈하였다. 또한 up counter이면 7-segment display에 U을, down counter이면 d을 띄워야 하므로 up이 1일 때는 digit[0]과 digit[6]에 0을, 나머지 값에 1을 설정하여 U을 표시할 수 있게 하였고, up이 0일 때는 digit[0]과 digit[5]에 0을, 나머지 값에 1을 설정하여 d을 표시할 수 있게 하였다. 그리고 7-segment display에 불을 들어오게 하는 변수인 dp는 OR연산으로 정의해주어 7개의 digit값 중 한 개라도 1을 갖는다면 불이 들어오도록 설정하였다. 이 코드로 잘 작동하는지 확인하기 위해 simulation 결과를 보면 초기화 변수인 rst가 0이고, up 변수가 1일 때 clk의 상승 엣지에서 출력은 초기화 값인 0000에서 0001, 0010, 0011, 0100 순으로 1씩 증가함을 볼 수 있다. 그러다가 up이 0이 된 순간 그 직후 clk의 상승 엣지에서는 down counter로 작동하여 1010, 1001, 1000, 0111, 0110, 0101순으로 1씩 작아지는 것을 확인할 수 있다. 이를 바탕으로 output table을 작성하면 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Clock Transitions | up | A | B | C | D |
| 초기 상태 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 | 1 |
| 4 | 1 | 0 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 | 0 | 1 |
| 6 | 1 | 0 | 1 | 1 | 0 |
| 7 | 1 | 0 | 1 | 1 | 1 |
| 8 | 1 | 1 | 0 | 0 | 0 |
| 9 | 1 | 1 | 0 | 0 | 1 |
| 10 | 1 | 1 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 | 0 | 1 |
| 12 | 0 | 1 | 0 | 0 | 0 |
| 13 | 0 | 0 | 1 | 1 | 1 |
| 14 | 0 | 0 | 1 | 1 | 0 |
| 15 | 0 | 0 | 1 | 0 | 1 |

이를 FPGA 동작으로도 확인하니 올바르게 동작함을 볼 수 있었다.

**4.**

이번 주 실험에서는 4-bit shift register, 4-bit ring counter, 4-bit up/down counter을 Verilog로 구현하고, simulation 결과를 확인한 뒤 FPGA의 동작을 살펴보았다. 먼저 4-bit shift register에서는 in로 들어온 입력 값이 첫번째 flip-flop의 출력이 되고, 이 출력 값은 두 번째 flip-flop의 출력이 되고, 두 번째 flip-flop의 출력은 세 번째 flip-flop의 출력이 되고 마지막으로 세 번째 flip-flop의 출력은 네 번째 flip-flop의 출력이 됨을 확인할 수 있었다. 4-bit ring counter에서는 초기값 0001부터 시작하여 1000, 0100, 0010, 0001순으로 1이 이동함을 FPGA로 볼 수 있었다. 4-bit up/down counter에서는 변수 up의 값에 따라 up counter로 작동할 지 down counter로 작동할 지 결정되고, 그 정보를 7-segment display에 띄운 뒤, FPGA을 통해 잘 작동함을 확인할 수 있었다.

**5.**

주파수 카운터란 전기적 진동수를 특정 시간을 설정하여 측정하기 위한 카운터이다. 보편적으로 1초를 기준으로 진동수를 측정하며 단위는 Hertz(Hz)을 사용한다. 입력 신호가 주파수 카운터의 입력 값으로 들어가면 카운터는 설정된 시간동안 입력 신호의 주기가 몇 번 들어오는지 세고, 이렇게 측정된 주기의 횟수는 주파수로 변환된다. 주파수 카운터는 라디오, 무선 통신이나 의학적 측정장치, 회로의 작동 주파수를 테스트하거나 측정할 때 주로 사용된다.

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

텍스트, 소프트웨어, 도표, 스크린샷이(가) 표시된 사진

자동 생성된 설명