13주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211522 이름: 김정환

**1.**

.................

Shift Register는 데이터가 하나씩 한 방향으로 이동하고 입력이 들어와서 저장되는 방식으로 구동하는 회로이다. 이는 회로도로 나타냈을 때는 아래와 같이 나타낼 수 있다.

도표, 라인, 직사각형, 평면도이(가) 표시된 사진

자동 생성된 설명

4-bit Shift Register를 Verilog로 구현하면 아래와 같이 나타낼 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위의 코드는 >> 연산자를 사용하여 4 bit인 out을 1씩 오른쪽으로 이동시킨다. 또한 rst가 1일 때는 0000으로 초기화한다. 이들은 모두 clk가 Rising edge일 때만 유효하게 동작한다. 이에 대한 결과를 확인하기 위해 작성한 Testbench 코드는 아래와 같다.

텍스트, 영수증, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

또한 Simulation 결과는 아래와 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

해당 경우를 살펴보면 처음 1이 들어오면 1000으로 저장되고, 다음 1이 들어왔을 때 1100으로 기존의 1이 오른쪽 한 칸 이동하고 1이 들어온다. 그 후에 0이 들어오면 0110으로 오른쪽 한 칸 이동 후 0이 들어오는 것을 볼 수 있다. 마지막으로는 0이 한 번 더 들어오면 0011으로 나오는 점을 볼 수 있다. 이를 표로 나타내면 아래와 같이 나타낼 수 있다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| clk | IN | L1 | L2 | L3 | L4 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 | 1 |

................

**2.**

.......................

Ring counter는 총 bit 내에 1이 하나만 유지되며 한 방향으로 순환하며 이동하는 형태의 출력을 통해 상태를 나타내는 counter이다. 이를 회로도로 나타내면 아래와 같이 나타낼 수 있다.

도표, 기술 도면, 평면도, 라인이(가) 표시된 사진

자동 생성된 설명

이를 Verilog 코드로 나타내면 아래와 같이 나타낼 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위의 코드는 clock이 Rising edge일 때 rst가 1이면 out을 1000으로 초기화하고, 이외에 clock이 들어올 때는 각각의 값들을 out[3]에 기존 out[0]의 값으로, out[2]에 기존 out[3]의 값으로 하는 등으로 밀어서 저장하도록 하였다. <=를 통해서 그때 그때 바로 저장하지 않고 한 번에 저장이 이루어지도록 함으로써 값이 꼬이지 않는다. 이 결과를 Simulation하기 위한 Testbench 코드는 아래와 같다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이를 통해 Simulation한 결과는 아래와 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

위의 Simulation을 확인하면 처음 rst가 1이고 clock이 들어왔을 때 out이 1000으로 초기화되는 것을 확인할 수 있다. 이 후 clock이 들어올 때마다 0100, 0010, 0001의 순서대로 상태가 변화하는 것을 확인할 수 있고, 0001에서 다시 clock이 들어오면 1000으로 돌아오는 것도 볼 수 있다. 이는 표로 나타내면 아래와 같이 나타낼 수 있다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clock | L1 | L2 | L3 | L4 |
| 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 |

이는 FPGA 보드에서도 확인을 진행하였는데 아래와 같은 constraints를 사용하였다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

이 때 rst에 연결한 버튼의 경우에는 Active low이므로 design 코드에서 rst에 ~을 붙이고 확인을 진행하였다. 이 경우에도 위의 Simulation과 동일한 결과가 나오는 것을 확인하였다.

.........................

**3.**

.......................

Up/Down counter는 이진수의 값이 하나씩 상승하며 상태를 변화시키는 counter와 하나씩 감소하며 상태를 변화시키는 counter가 모두 기능하도록 구현한 counter이다. 이는 회로도로 나타내면 아래와 같이 나타낼 수 있다.

도표, 평면도, 기술 도면, 개략도이(가) 표시된 사진

자동 생성된 설명

이를 Verilog 코드로 구현하면 아래와 같이 나타낼 수 있다.

텍스트, 스크린샷, 폰트, 영수증이(가) 표시된 사진

자동 생성된 설명

위의 코드는 if문과 clock을 위한 always문으로 구성하였다. flag의 값이 1일 때는 Up counter가 되도록 하였고, 이외에는 Down counter이다. 따라서 clock이 들어올 때 rst가 1이면 flag가 1일 때는 0000으로, 이외에는 1111으로 초기화하도록 구현했다. 또한 flag가 1일 때는 out에 1을 더해주며 상태를 변화시키고, 이외에는 out에서 1을 빼며 상태를 변화시킨다. sett과 DS는 강의자료에 주어진 대로 7-segment display로 모드를 나타내는데 사용하였다. DS에는 해당 모양에 맞게 값을 저장하도록 하였고, sett는 1로 고정하여 display가 보드에서 켜지도록 한다. 결과를 확인하기 위한 Testbench 코드는 아래와 같다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이를 통해 Simulation한 결과는 아래와 같다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

위의 결과를 살펴보면 우선 flag가 1인 경우에 rst가 1이면 0000으로 초기화되는 것을 볼 수 있다. 이 후에 clock이 들어올 때마다 0001, 0010, 0011, 0100, … , 1111까지 증가한 후에 다시 0000으로 돌아가는 결과를 볼 수 있다. 다음으로 flag가 0인 경우 rst가 1이면 1111으로 초기화되는 것을 볼 수 있다. 이 후에 clock이 들어올 때마다 1110, 1101, 1100, 1011, …, 0000까지 간 후에 다시 1111으로 돌아가는 것을 확인할 수 있다. 일부만 표로 나타내면 아래와 같이 나타낼 수 있다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Clock | L1 | L2 | L3 | L4 | DISPLAY |
| 1 | 0 | 0 | 0 | 0 | U |
| 2 | 0 | 0 | 0 | 1 | U |
| 3 | 0 | 0 | 1 | 0 | U |
| 4 | 0 | 0 | 0 | 1 | d |
| 5 | 0 | 0 | 0 | 0 | d |
| 6 | 1 | 1 | 1 | 1 | d |

이는 FPGA 보드에서도 확인을 진행하였는데 아래와 같은 constraints를 사용하였다.

텍스트, 스크린샷, 보라색, 바이올렛색이(가) 표시된 사진

자동 생성된 설명

이 역시도 2번과 같이 rst에 대해서는 보드에서 확인 시에는 ~로 반대로 바꿔주는 작업을 거쳤다. 이 역시도 위의 Simulation과 동일한 결과를 확인할 수 있었고 중간에 Up Down을 바꾸는 경우에 대해서도 잘 동작하는 점을 확인하였다.

.........................

**4.**

.......................

본 실험에서는 4-bit Shift Register, 4-bit Ring counter와 4-bit Up/Down counter가 어떻게 동작하는지 파악하고 Verilog를 통해서 구현한 후 FPGA에서 확인하였다. 4-bit Shift Register는 >> 연산자를 사용하여 구현하였다. 이를 통해서 한 칸씩 오른쪽으로 Shift를 진행한 후에 제일 왼쪽 bit에 들어온 input을 넣는 방식을 사용했다. 다음으로 4-bit Ring counter의 경우에는 각각의 값을 =가 아닌 <=로 저장하도록 하여 값의 저장이 한 번에 이루어지도록 하여 값이 한 칸씩 돌도록 하였다. 이에 따라 전체 4 bit에서 1은 하나로 유지되며 계속 순환하는 것을 볼 수 있다. 마지막으로 4-bit Up/Down counter는 flag 변수를 통해서 Up과 Down을 구분하였고, 8주차에서 사용한 7-segment Display를 사용하여 각각을 표시하도록 하였다. 이외에는 이전 주차의 counter와 유사하게 +와 -로 연산하는 것을 통하여 구현하였다.

.........................

**5.**

.......................

지금까지 배운 counter를 응용하여 FPGA에서 1초를 측정할 수 있도록 하는 방법에 대해서 찾아보았다. 이는 clock마다 들어오는 만큼 counter를 통해서 세는 점이 중요한데, 사용하는 FPGA의 칩에서 사용하는 System Clock에 대한 정보가 필요하다. 실습에서 사용하는 보드의 경우에는 100MHz의 System Clock을 가지고 있다. 즉 100,000,000의 clock이 1초에 들어오는 것임을 의미하고 이는 이진수 변환 시에 27자리의 이진수가 나오는 것을 확인할 수 있다. 따라서 해당 counter는 27자리의 이진수로 count할 수 있도록 구현해야 하고, 이 수만큼 셌을 때가 1초가 됨을 알 수 있다.

텍스트, 영수증, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명

위의 예시로는 125MHz인 보드 기준으로 1초가 지나는 것을 판단하는 코드의 부분임을 볼 수 있다. 이를 사용하면 시계, 스톱워치 등을 구현할 수 있다.

.........................

출처

<https://www.allaboutcircuits.com/textbook/digital/chpt-11/synchronous-counters/>

<https://sytjsdid.tistory.com/3>