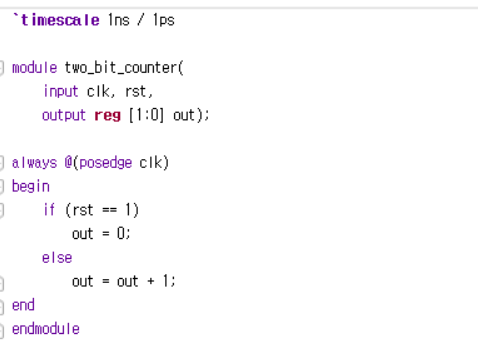
12주차 결과보고서

전공: 물리학과 학년: 3학년 학번: 20191286 이름: 김나현

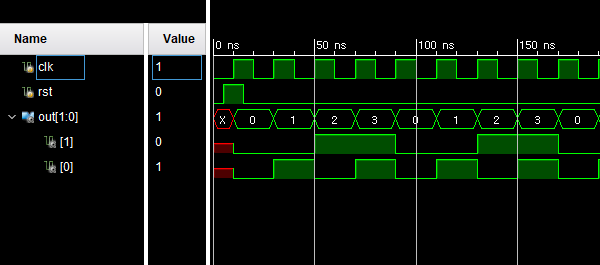
1. 2-bit counter의 결과 및 simulation 과정에 대해서 설명하시오(verilog source, 출력 예시, 과정을 상세히 적을 것).

클럭 펄스가 있을 때마다 00, 01, 10, 11, 00을 반복하는 2-bit binary counter를 만들기 위해 다음 <Figure 1>과 같이 Verilog 코딩하였습니다.



<Figure 1> two\_bit\_counter.v

이 Verilog 코드는 clk 신호가 상승 에지일 때, rst가 1이면 out이 0이 되고 rst가 1이 아니면 out이 out+1가 되게 하는 코드인데 rst는 counter를 초기화해주는 입력이고, out이 하나씩 증가하는 것은 00, 01, 10, 11, 00가 반복되도록 하는 것입니다. 따라서 test bench를 만들어 해당 Verilog 코드의 simulation 결과를 확인해보면 다음 <Figure 2>와 같습니다.



<Figure 2> two\_bit\_counter.v의 simulation 결과

rst가 1일 때 clk이 상승 에지이면 out[0]과 out[1]는 0으로 세팅되고 두 번째 clk의 상승 에지에서는 rst가 1이 아니므로 01이 되고 그 다음 clk의 상승 에지마다 10, 11, 00, 01을 반복하게 됩니다.

텍스트이(가) 표시된 사진

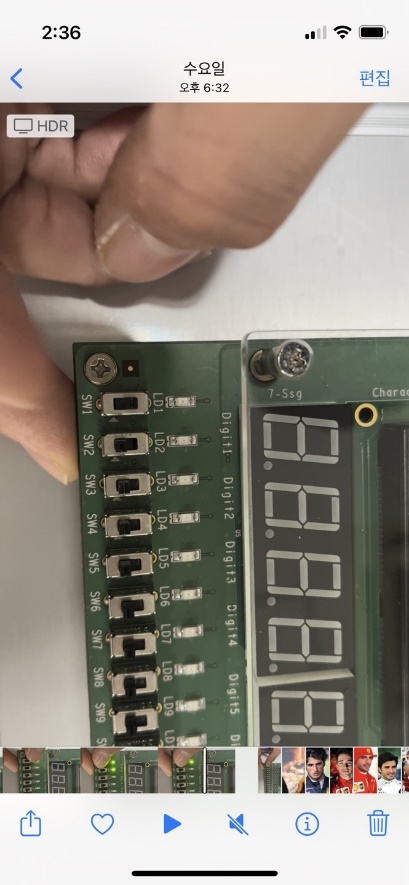
자동 생성된 설명FPGA를 이용하여 2-bit binary counter를 Verilog로 구현한 것이 제대로 돌아가는지 확인하기 위해 다음 <Figure 3>과 같이 const.xdc 소스를 만들어 연결합니다.

<Figure 3> const.xdc

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명FPGA를 이용하여 2-bit binary counter의 동작을 확인하면 다음 <Figure 4>와 같습니다.

<Figure 4> FPGA를 통해 확인한 2-bit binary counter의 동작

rst를 의미하는 SW2를 1로 바꾸었다가 다시 0으로 바꾼 후, clk 펄스를 의미하는 SW1을 1로 바꾸면 1로 바꾸는 순간, out[0]을 의미하는 LD2에 불이 들어오고 SW1을 다시 0으로 바꿨다가 1로 바꾸면 LD2의 불이 꺼지고 out[1]을 의미하는 LD1에 불이 들어옵니다. 다시 SW1을 0으로 바꿨다가 1로 바꾸면 LD1과 LD2의 불이 모두 들어오는 것을 볼 수 있고, SW1을 0으로 바꿨다가 1로 바꾸면 LD1과 LD2의 불이 모두 꺼지고 SW1을 0으로 바꿨다가 1로 바꾸는 것을 반복하면 <Figure 4>가 반복적으로 나타나는 것을 확인할 수 있습니다. SW1을 0에서 1로 바꾸는 순간, LED의 불이 들어오는 것이 변한다는 사실을 통해 verilog 코드에서 clk이 상승일 때마다 always문으로 수행한다는 것이 잘 적용된 것을 관찰할 수 있었습니다. 또한 rst 입력을 0에서 1로 바꿨다가 다시 0으로 바꿈으로써 2-bit binary counter가 00에서부터 시작할 수 있도록 verilog 코딩한 것이 FPGA를 통해서 관찰할 때도 잘 적용되어 나타난다는 것을 확인할 수 있었습니다.

1. 4-bit decade counter의 결과 및 simulation 과정에 대해서 설명하시오(verilog source, 출력 예시, 과정을 상세히 적을 것).

2-bit binary counter가 00, 01, 10, 11을 반복한다면 4-bit binary counter는 0000부터 1111을 반복할 것입니다. 하지만 4-bit decade counter는 4비트 이진수로 나타낼 수 있는 16가지 상태를 모두 나타내는 4-bit binary counter와 달리 16가지 상태 중에서 0부터 9까지 10가지의 상태만을 나타내기 위해 0000부터 1001을 반복하는 counter입니다. 클럭 펄스가 있을 때마다 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001을 반복하는 4-bit decade counter를 만들기 위해 다음 <Figure 5>과 같이 Verilog 코딩하였습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<Figure 5> four\_bit\_decade\_counter.v

이 Verilog 코드는 clk 신호가 상승 에지일 때, rst가 1이면 out이 0이 되고 rst가 1이 아니고 out이 9이면 out이 다시 4’b000이 되며 rst가 1이 아니고, out이 9도 아니면 out이 out+1가 되게 하는 코드인데 여기서도 마찬가지로 rst는 counter를 초기화해주는 입력이고, out이 하나씩 증가하는 것은 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001가 반복되도록 하는 것입니다. 따라서 test bench를 만들어 해당 Verilog 코드의 simulation 결과를 확인해보면 다음 <Figure 6>과 같습니다.

텍스트, 점수판이(가) 표시된 사진

자동 생성된 설명

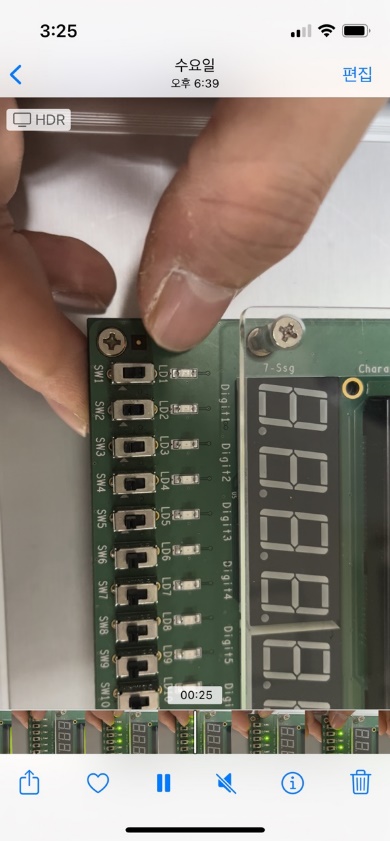
<Figure 6> four\_bit\_decade\_counter.v의 simulation 결과

rst가 1일 때 clk이 상승 에지이면 out[0]과 out[1], out[2], out[3]는 모두 0으로 세팅되고 두 번째 clk의 상승 에지에서는 rst가 1이 아니므로 0001이 되고 그 다음 clk의 상승 에지마다 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001을 반복하게 됩니다.

텍스트이(가) 표시된 사진

자동 생성된 설명FPGA를 이용하여 4-bit decade counter를 Verilog로 구현한 것이 제대로 돌아가는지 확인하기 위해 다음 <Figure 7>과 같이 const.xdc 소스를 만들어 연결합니다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명<Figure 7> const.xdc

FPGA를 이용하여 4-bit decade counter의 동작을 확인하면 다음 <Figure 8>과 같습니다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기, 자동판매기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명<Figure 8> FPGA를 통해 확인한 4-bit decade counter의 동작

rst를 의미하는 SW2를 1로 바꾸었다가 다시 0으로 바꾼 후, clk 펄스를 의미하는 SW1을 1로 바꾸면 1로 바꾸는 순간, out[0]을 의미하는 LD4에 불이 들어오고 SW1을 다시 0으로 바꿨다가 1로 바꾸면 LD2의 불이 꺼지고 out[1]을 의미하는 LD3에 불이 들어옵니다. 다시 SW1을 0으로 바꿨다가 1로 바꾸면 LD3과 LD4의 불이 모두 들어오는 것을 볼 수 있고, SW1을 0으로 바꿨다가 1로 바꾸면 LD3과 LD4의 불이 모두 꺼지고 LD2에 불이 들어오며 SW1을 0으로 바꿨다가 1로 바꾸는 것을 반복하면 <Figure 8>의 순서대로 나타나는 것을 확인할 수 있습니다. 마찬가지로 SW1을 0에서 1로 바꾸는 순간, LED의 불이 들어오는 것이 변한다는 사실을 통해 verilog 코드에서 clk이 상승일 때마다 always문으로 수행한다는 것이 잘 적용된 것을 관찰할 수 있었습니다. 또한 rst 입력을 0에서 1로 바꿨다가 다시 0으로 바꿈으로써 4-bit decade counter가 0000에서부터 시작할 수 있도록 verilog 코딩한 것이 FPGA를 통해서 관찰할 때도 잘 적용되어 나타난다는 것을 확인할 수 있었습니다.

1. 4-bit 2421 decade counter의 결과 및 simulation 과정에 대해서 설명하시오(verilog source, 출력 예시, 과정을 상세히 적을 것).

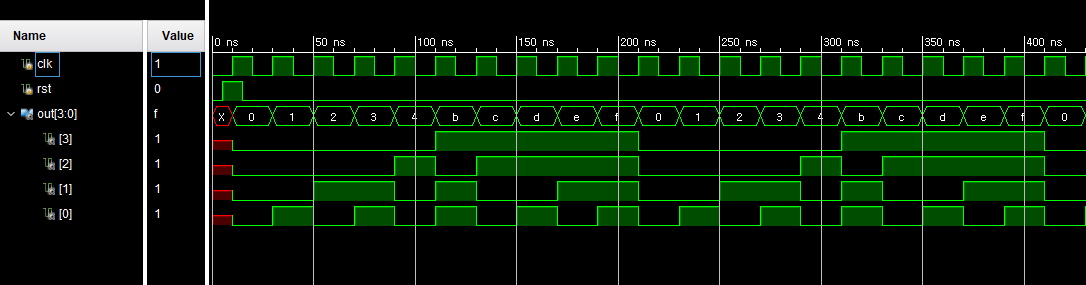
4-bit decade counter가 0000부터 1001을 반복한다면 4-bit 2421 decade counter는 5를 1011로, 6을 1100, 7을 1101, 8을 1110, 9를 1111로 표현하는 2421 code를 적용하여 0000부터 1111 중 0101, 0110, 0111, 1000, 1001, 1010을 건너뛰고 나타내는 counter입니다. 클럭 펄스가 있을 때마다 0000, 0001, 0010, 0011, 0100, 1011, 1100, 1101, 1110, 1111을 반복하는 4-bit 2421 decade counter를 만들기 위해 다음 <Figure 9>와 같이 Verilog 코딩하였습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<Figure 9> four\_bit\_2421\_decade\_counter.v

이 Verilog 코드는 clk 신호가 상승 에지일 때, rst가 1이면 out이 0이 되고 rst가 1이 아니고 out이 4이면 out는 4’b1011이 되며 rst가 1이 아니고, out이 4도 아니면 out이 out+1가 되게 하는 코드인데 여기서도 마찬가지로 rst는 counter를 초기화해주는 입력이고, out이 하나씩 증가하는 것은 0000, 0001, 0010, 0011, 0100, 1011, 1100, 1101, 1110, 1111가 반복되도록 하는 것입니다. 따라서 test bench를 만들어 해당 Verilog 코드의 simulation 결과를 확인해보면 다음 <Figure 10>과 같습니다.



<Figure 10> four\_bit\_decade\_counter.v의 simulation 결과

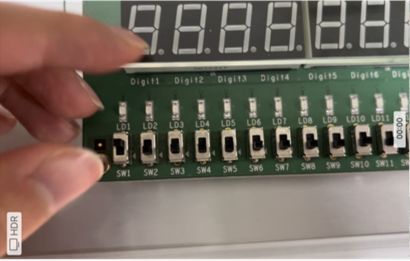
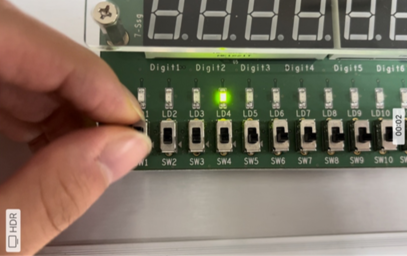
rst가 1일 때 clk이 상승 에지이면 out[0]과 out[1], out[2], out[3]는 모두 0으로 세팅되고 두 번째 clk의 상승 에지에서는 rst가 1이 아니므로 0001이 되고 그 다음 clk의 상승 에지마다 0010, 0011, 0100, 1011, 1100, 1101, 1110, 1111을 반복하게 됩니다.

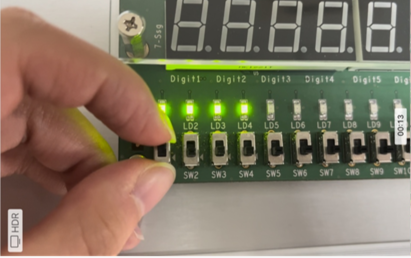
FPGA를 이용하여 4-bit decade counter를 Verilog로 구현한 것이 제대로 돌아가는지 확인하기 위해 다음 <Figure 11>과 같이 const.xdc 소스를 만들어 연결합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<Figure 11> const.xdc

FPGA를 이용하여 4-bit decade counter의 동작을 확인하면 다음 <Figure 12>와 같습니다.

<Figure 12> FPGA를 통해 확인한 4-bit 2421decade counter의 동작

rst를 의미하는 SW2를 1로 바꾸었다가 다시 0으로 바꾼 후, clk 펄스를 의미하는 SW1을 1로 바꾸면 1로 바꾸는 순간, out[0]을 의미하는 LD4에 불이 들어오고 SW1을 다시 0으로 바꿨다가 1로 바꾸면 LD2의 불이 꺼지고 out[1]을 의미하는 LD3에 불이 들어옵니다. 다시 SW1을 0으로 바꿨다가 1로 바꾸면 LD3과 LD4의 불이 모두 들어오는 것을 볼 수 있고, SW1을 0으로 바꿨다가 1로 바꾸면 LD3과 LD4의 불이 모두 꺼지고 LD2에 불이 들어오며 SW1을 0으로 바꿨다가 1로 바꾸는 것을 반복하면 <Figure 8>의 순서대로 1011, 1011, 1100, 1101, 1110, 1111가 나타나는 것을 확인할 수 있습니다. 마찬가지로 SW1을 0에서 1로 바꾸는 순간, LED의 불이 들어오는 것이 변한다는 사실을 통해 verilog 코드에서 clk이 상승일 때마다 always문으로 수행한다는 것이 잘 적용된 것을 관찰할 수 있었습니다. 또한 rst 입력을 0에서 1로 바꿨다가 다시 0으로 바꿈으로써 4-bit decade counter가 0000에서부터 시작할 수 있도록 verilog 코딩한 것이 FPGA를 통해서 관찰할 때도 잘 적용되어 나타난다는 것을 확인할 수 있었습니다.

1. 결과 검토 및 논의 사항

출력 out의 모든 자릿수를 0으로 세팅해주는 리셋 입력 rst를 사용하지 않으면 임의의 값에서부터 카운터가 시작되기 때문에 rst를 사용하여 카운터의 기본값인 00이나 0000에서부터 시작되도록 하였고 clk 신호가 posedge, 즉 상승 에지일 때마다 Verilog 코드의 always문을 수행하도록 하여 입력 clk이 0에서 1로 바뀌는 순간 출력이 바뀌도록 하였습니다. 00, 01, 10, 11을 단순히 반복하는 2-bit binary counter나 0000부터 1111을 반복하는 4-bit binary counter의 경우, always문에서 rst가 1인 경우를 제외하고는 항상 out을 1씩 증가시키면 되지만 4-bit decade counter나 4-bit 2421 decade counter는 현재 출력이 9이거나 4이면 0000이나 1011으로 out에 1을 더하는 것이 아닌 중간 값들을 건너뛰는 과정이 필요하므로 if, else if 문 등을 활용하여 현재 출력에 따라 다음 출력 값이 어떻게 출력될 것인지를 결정해줘야 합니다. Simulation 결과나 FPGA를 통해 확인한 동작 모두 2-bit binary counter, 4-bit decade counter, 4-bit 2421 decade counter가 제대로 동작하고 있음을 보여주므로 위와 같이 코딩한 Verilog 코드가 올바르다는 것을 알 수 있었습니다.

1. 추가 이론 조사 및 작성

Verilog 코딩을 할 때, output으로 out1, out2, out3, out4와 같이 사용하지 않고 [3:0] out처럼 사용하여 한 번에 나타낼 수 있었습니다. 이때, [3:0] out과 같이 사용하면 위의 <Figure 9>와 같이 out4, out3, out2, out1으로 0, 0, 0, 0을 출력하고 싶다면 out4=0; out3=0; out2=0; out1=0; 대신 out=4’b0000;이라고 단순화하여 표현할 수 있다는 장점이 있습니다. 또한, always나 initial과 같은 절차형 할당문을 통해 값을 받는 객체는 reg라는 register type으로 나타내야 하므로 always문에서 결정되는 out은 output reg [3:0] out;이라고 선언하였습니다.