컴퓨터공학실험2 3주차 결과 보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20201635 이름: 전찬

**0. 목차**

1. FPGA 동작법 설명

2. 3-input AND gate의 simulation 결과 및 과정 설명

3. 4-input AND gate의 simulation 결과 및 과정 설명

4. 3-input OR gate의 simulation 결과 및 과정 설명

5. 4-input OR gate의 simulation 결과 및 과정 설명

6. 결과 검토 및 논의 사항

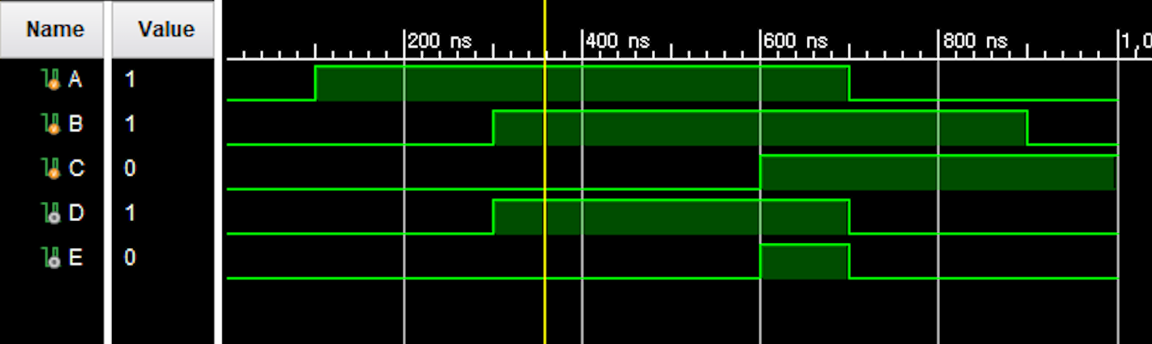
7. 추가 이론 조사 및 작성

**1. FPGA 동작법**

실습에서 직접 진행하지 않았지만, Verilog 동작 과정은 존재하는 프로젝트에 xdc 파일을 추가하며 진행한다. xdc 파일을 통해서 실제 FPGA상의 Pin과 Verilog에서 논리적으로 존재하는 port를 “set\_property -dict {PACKAGE\_PIN G21 IOSTANDARD LVCMOS33} [get\_ports a]” 와 같은 형태로 적용할 수 있다. 이후에는 Run Synthesis, Run Implementation, Generate Bitstream 순으로 진행하며 Verilog에서 코딩한 프로젝트를 디버깅하며, FPGA에 적용 가능한 파일로 변환해줄 수 있다.

**2. 3-input AND gate의 simulation 결과 및 과정**

3-input AND 게이트는 2-input AND를 응용해서(3-input을 2-input으로 decomposition) 만들 수 있다. 3가지의 input 중 두 input을 먼저 선택해서 2-input AND를 적용한 이후, 나온 결과를 나머지 한 input과 다시 2-input AND를 적용하면 된다. 이렇게 만들어낸 3-input AND gate의 simulation 결과는 아래와 같다.( A, B, C는 input이며, A&B=D, C&D=E 이다.)



<3-input AND gate simulation>

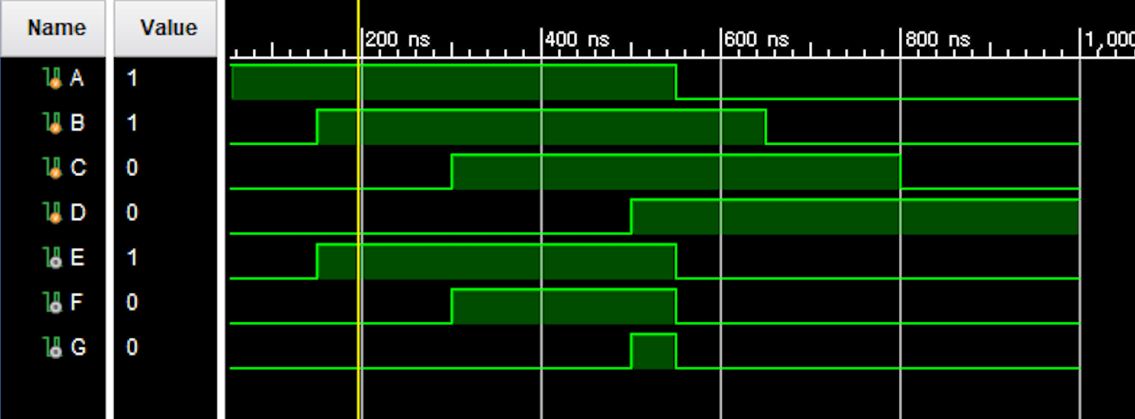
또한 3-input AND gate의 진리표는 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3-input AND Gate | | | | |
| input | | | output | |
| A | B | C | D | E |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

<3-input AND gate 진리표>

**3. 4-input AND gate의 simulation 결과 및 과정**

4-input AND 게이트 또한 3-input AND gate와 비슷하게 2-input AND를 응용해서 만들 수 있다. 4가지의 input 중 두 input을 먼저 선택해서 2-input AND를 적용한 이후, 나온 결과를 나머지 두 input중 한 input과 다시 2-input AND를 적용한다. 이 과정을 다시 한 번 반복하면 4-input AND gate를 만들어낼 수 있으며, 이렇게 만들어낸 4-input AND gate의 simulation 결과는 아래와 같다.( A, B, C, D는 input이며, A&B=E, C&E=F, F&D=G 이다.)



<4-input AND gate simulation>

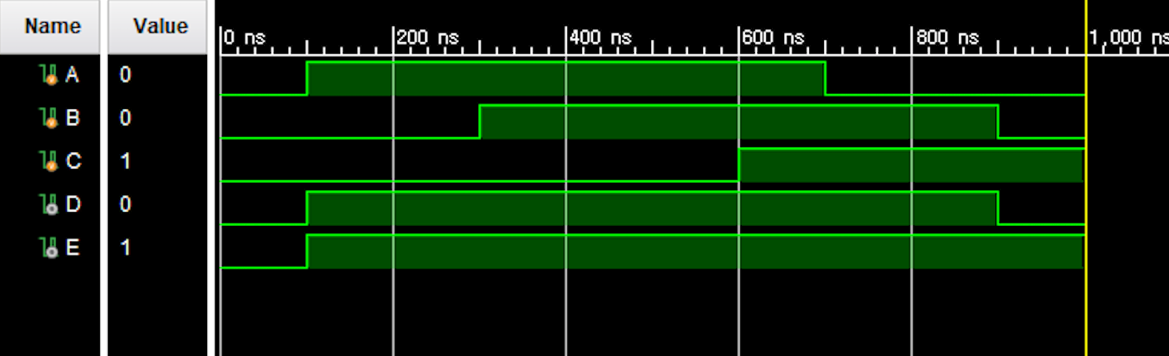
또한 4-input AND gate의 진리표는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4-input AND gate | | | | | | |
| input | | | | output | | |
| A | B | C | D | E | F | G |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

<4-input AND gate 진리표>

**4. 3-input OR gate의 simulation 결과 및 과정**

3-input OR 게이트 또한 위에서 여러 input을 갖는 AND gate를 만들어낸 과정과 동일하게 만들어낼 수 있다. 3가지의 input 중 두 input을 먼저 선택하고 2-input OR gate를 적용하고, 이후 나머지 하나의 input과 앞에서 나온 output을 다시 2-input OR 적용해주면 된다. simulation 결과는 아래와 같다..( A, B, C는 input이며, A|B=D, C|D=E 이다.)



<3-input OR gate simulation>

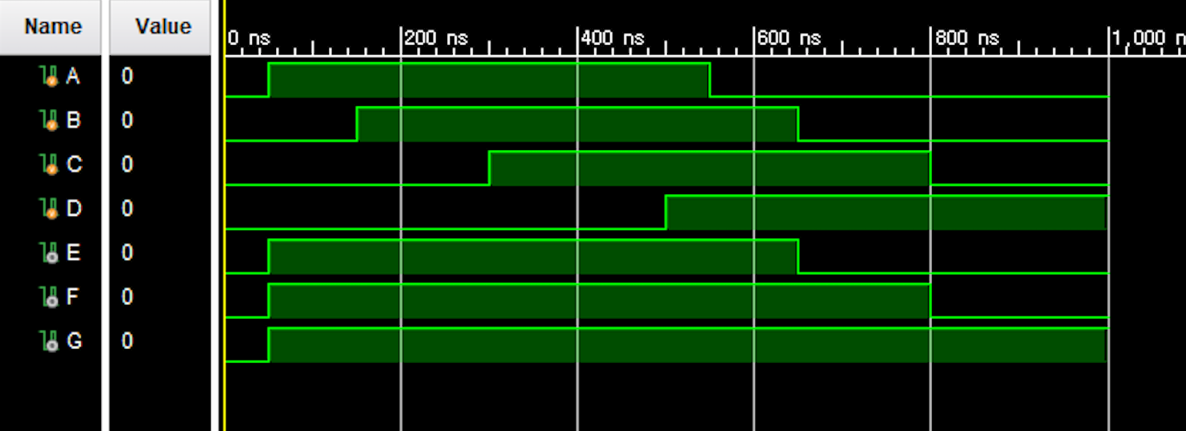
또한 3-input OR gate의 진리표는 아래와 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3-input OR Gate | | | | |
| input | | | output | |
| A | B | C | D | E |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 |

<3-input OR gate 진리표>

**5. 4-input OR gate의 simulation 결과 및 과정**

4-input OR gate 또한 위의 여러 gate들의 제작 과정과 동일하게 제작한다. 4가지의 input 중 두 input을 먼저 선택해서 2-input OR을 적용한 이후, 나온 결과를 나머지 두 input중 한 input과 다시 2-input OR을 적용한다. 이 과정을 다시 한 번 반복하면 4-input OR gate를 만들어낼 수 있으며, 이렇게 만들어낸 4-input AND gate의 simulation 결과는 아래와 같다.( A, B, C, D는 input이며, A|B=E, C|E=F, F|D=G 이다.)



<4-input OR gate simulation>

또한 4-input OR gate의 진리표는 다음과 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4-input AND gate | | | | | | |
| input | | | | output | | |
| A | B | C | D | E | F | G |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

<4-input OR gate 진리표>

**6. 결과 검토 및 논의 사항**

위 진리표들을 통해, 실습에서 제작한 3-input, 4-input gate의 simulation 결과를 토대로 여러 input을 갖는 AND/OR gate를 2-input으로 분해해서 생각해도 문제 없음을 파악할 수 있다. 하지만 실습에서 simulation을 진행할 때, 진리표에는 모든 경우가 고려되어 있지만, simulation에서는 볼 수 없는 경우가 존재해서 input의 설계를 더 정확하게 하면 더 정확한 결과를 얻어낼 수 있었을 것 같다.(4-input OR gate에서 A = 1, B = 1, C = 0, D = 1인 경우가 simulation에서는 표현되어 있지 않다.)

**7. 추가 이론 조사 및 작성**

위에서 2-input AND, OR gate를 활용해 더 많은 input을 가질 수 있는 gate들을 만들어냈다. 이번 실습과 비슷하게 대부분의 gate들은 AND, OR, NOT gate를 바탕으로 만들어낼 수 있다. 이때 수식을 작성할 때, 곱셈이 덧셈에 우선순위인 것을 아는 것처럼, AND, OR, NOT gate의 우선 순위를 알고 있다면 여러 게이트를 더 짧은 코드로 만들 수 있다. 결론 먼저 말하자면, NOT -> AND -> OR 순으로 우선 순위가 결정되어 있으며, 간단한 Verilog 코딩을 통해서 알아낼 수 있다.