7주차 예비보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20211547 이름: 신지원

**1.**

Adder 와 subtractor, 그리고 code converter 의 개념을 이해하고 Verilog 코드를 짠 뒤, FPGA로 simulation을 확인해보고자 한다. 카르노맵을 통해 논리식을 직접 간소화하며 코드를 짜며 진리표를 작성해보고자 한다. 나아가 실습 때 도출한 schematic 도 첨부하여 회로의 흐름을 자세히 이해한다.

**2.**

half adder 를 구현하기 위해서 sum 과 carry 의 약자인 s 와 c 를 output 으로 선언하였다. Verilog 코드로는 각각 assign s = a^b; assign c = a&b; 로 선언하였다.

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

simulation 으로 나온 결과는 위와 같다. carry 는 a와 b가 둘다 1일 때만 1을 출력하는데 시뮬레이션 상에서도 이러한 사실이 잘 구현되었다.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| a | b | s | c |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

시뮬레이션을 바탕으로 작성한 진리표다. s 를 a와 b의 XOR gate 의 출력값으로 작성해주었기 때문에 더한 다음의 해당 자리 수를 나타내며 c 는 올림값을 나타낸다.

full adder 를 구현하기 위해서 half-adder 와 마찬가지로 sum 과 carry 의 약자인 s 와 c 를 output 으로 선언하였는데 앞선 half-adder 와 다른 점은 입력이 3개기 때문에 마지막 하나를 cin 으로 출력을 c가 아닌 cout 으로 선언하였다는 것이다. 세 개의 입력의 가산기 코드를 작성하기 위해선 assign s = a^b^cin; 로 작성하여 1이 홀수일 때만 1을 출력하도록 하였다. 또한 카르노맵을 사용하여 assign cout = ( cin & (a^b))|(a&b); 의 논리식을 도출하였다.

스크린샷, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

full-adder 의 simulation 은 input 중 하나 혹은 세 개의 신호가 1일 때만 s가 1을 가지며, 두 개의 신호가 1일 때는 0을 갖는다. cout은 두 개와 세개의 신호가 1일 때만 1을 갖는다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| A | B | Cin | C | Cout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

simulation 을 바탕으로 작성한 진리표다. 논리식을 구한 이론을 바탕으로 계산하였을 때도 같은 결과가 나온다.

**3.**

half subtracter 를 구현하기 위해서 difference 과 borrow 의 약자인 d 와 b 를 output 으로 선언하였다. Verilog 코드로는 각각 assign D = a^b; assign B = (~a)&b; 로 선언하였다.

스크린샷, 컴퓨터, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

half-subtracter 의 simulation 의 D는 (1,0), (0,1) 에서만 1을 출력한다. 또한 B는 빼는 수가 더 클 때만 1을 출력한다.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| A | B | D | B |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

simulation 을 바탕으로 작성한 진리표다. 논리식을 구한 이론을 바탕으로 계산하였을 때도 같은 결과가 나온다.

full subtractor 를 구현하기 위해서 half- subtractor 와 마찬가지로 difference 과 borrow 의 약자인 d 와 b 를 output 으로 선언하였는데 앞선 half- subtractor 와 다른 점은 입력이 3개기 때문에 마지막 하나를 bn\_1 으로 출력을 d가 아닌 dn 으로, b가 아닌 bn으로 선언하였다는 것이다. 세 개의 입력의 감산기 코드를 작성하기 위해선 각각 assign Dn = (a^b)^bn\_1; assign Bn = ( ~(a^b) & bn\_1) | (~a & b); 로 작성하였다. 특히 가산기보다 감산기가 입력의 순서가 중요하기 때문에 괄호로 잘 닫아준 모습이다.

스크린샷, 소프트웨어, 텍스트, 디스플레이이(가) 표시된 사진

자동 생성된 설명

 full-subtracter 의 simulation 의 Dn는 홀수개의 input이 1의 신호를 가질 때만 1을 출력한다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| An | Bn | Bn-1 | Dn | Bn |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

simulation 을 바탕으로 작성한 진리표다. 논리식을 구한 이론을 바탕으로 계산하였을 때도 같은 결과가 나온다.

4.

텍스트, 번호, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명 텍스트, 번호, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

SOP = A + BD + BC SOP = A + BC’D’ + BC

POS = ( A + C + D )( A + B ) POS = ( A + B )( A + C + D’ )

텍스트, 번호, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명 텍스트, 번호, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

SOP = A + BC’D + A’B’C SOP = D

POS = ( A + B + C )(A + C + D)(B’ + C’) POS = D

위에서 구현한 4개의 카르노맵으로 입력이 a, b, c, d 고, 출력이 w, x, y, z 인 코드를 구현하였다. SOP 를 기준으로 구현하였으며 assign w = a | ( b & d ) | ( b & c); assign x = a | ( b & c ) | ( b & ~d); assign y = a | ( ~b & c ) | ( d & b & ~c ); assign z = d; 로 선언하였다.

스크린샷, 디스플레이, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Input | | | | Output | | | |
| A | B | C | D | X | Y | Z | W |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

simulation 을 토대로 진리표를 작성하였다. 10진수로 5를 의미하는 0101은 그대로 0101이 아니라 1011로 나타난 이유는5가 4와 1의 보수 관계를 표현하기 위해 4인 0100 의 보수로 나타났기 때문이다.

5.   결과 검토 및 논의사항.

half-adder 일 경우보다 full-adder 일 때가 더 복잡한 논리식이 나왔다. 카르노맵을 사용하여 논리식을 작성하는 것도 좋지만, 단순한 조건문으로 작성해도 sum 과 carry 를 표현할 수 있을 것이다. 만약 3개의 input 이라면 carry 를 표현할 때 입력의 신호가 2개이상 1일 때 무조건 1을 출력하도록 할 수 있을 것이다. half-subtracter 의 경우에도, a보다 b가 더 클때 b가 발생하는 조건문으로 코드를 구현할 수 있을 것이다. 아래는 각 감/가산기의 schematic 다.

텍스트, 스크린샷, 도표, 디스플레이이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진

자동 생성된 설명

adder 의 schematic

텍스트, 스크린샷, 도표, 디스플레이이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

subtracter 의 schematic

**6.**

배열 승산기( Array Multiplier )는 디지털 논리회로에서 곱셈을 구하는 여러 방법 중 하나다. 두 수가 있다면 각 자릿수끼리의 부분곱을 자릿수에 맞추어 배열을 만들고 배열의 같은 열끼리 더한다. 이때, 올림수가 발생한다면 올림수를 올리며 진행한다. 하지만 논리곱 게이트의 수가 급수적으로 늘어날 수 있다는 문제점이 발생할 수 있다.