6주차 결과보고서

전공: 아트엔테크놀로지 학년: 3학년 학번: 20191172 이름: 함승우

1.실험 목적

반가산기와 전가산기의 구현 과정과 반감산기, 전감산기의 구현 과정을 살펴보고, FPGA를 통해서 논리식과 일치한지, 알맞게 코딩했는지 확인한다. 또한, 8421 code를 2421 code로 convert해주는 논리회로를 설계하고, Truth-table로 카르노 맵을 작성하여 논리식을 간소화하여 표현한다.

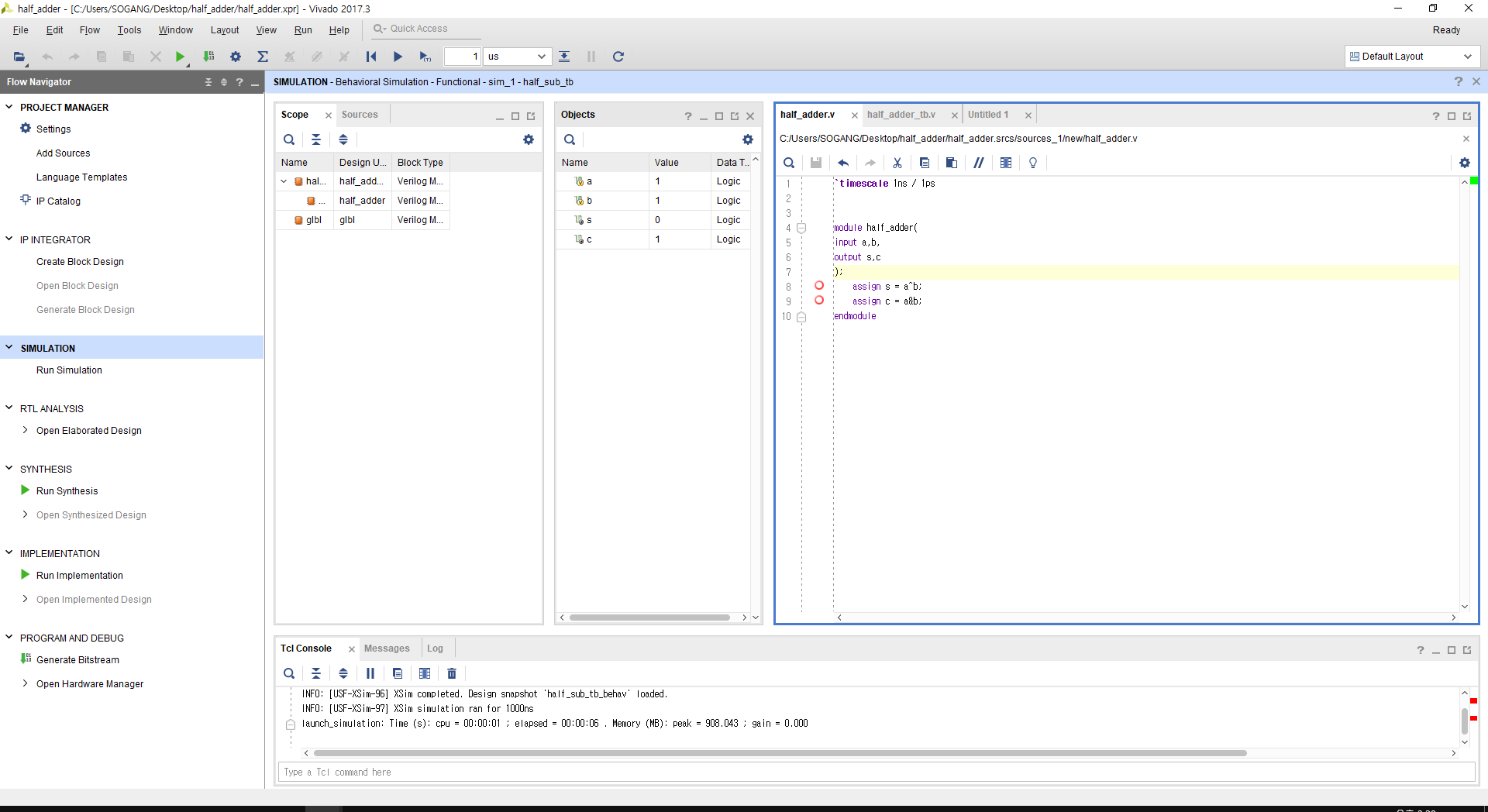
2.Full Adder 및 Half adder의 simulation 결과 및 과정에 대해서 설명하시오.(진리표 포함).

Half Adder의 경우에는 2개의 입력과 출력을 가진다. 입력은 1-bit의 수 2개, 출력은 이 입력값들의 합인 Sum, 그리고 자리올림의 값을 가지는 Carry를 가진다. Sum과 Carry를 구현하기 위해서는 XOR, AND 연산을 통해서 구현한다.

도표, 스케치, 그림, 라인 아트이(가) 표시된 사진

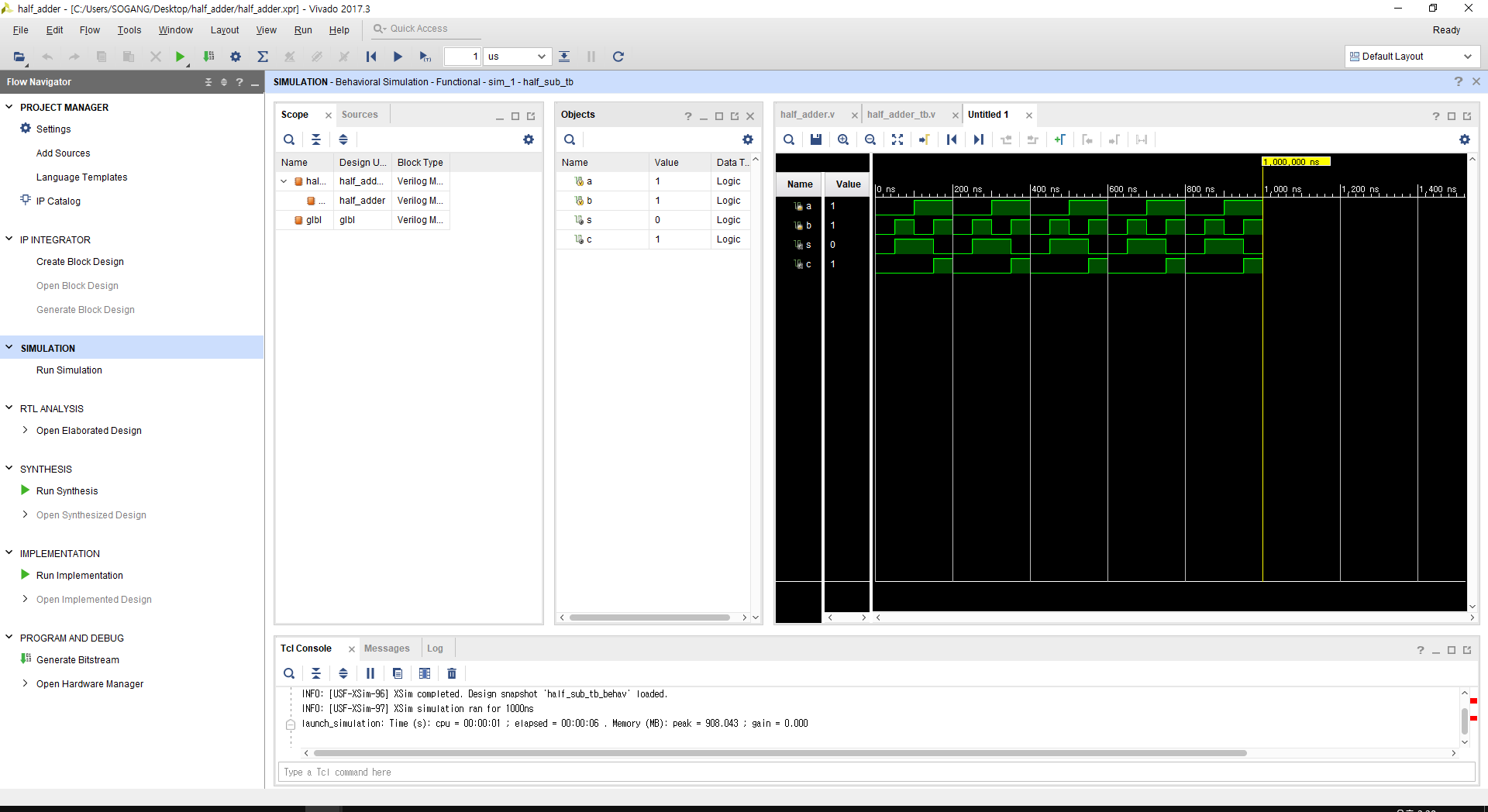
자동 생성된 설명

* Half Adder의 구조를 (a)로 Verilog 코딩한 결과



출력 값인 Sum의 경우에는 a와 b를 XOR 연산한 값을, Carry인 C값은 a와 b를 AND 연산을 하여 결정한다.

* Verilog의 simulation 결과와 진리표



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | S | C | a^b | a&b |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |

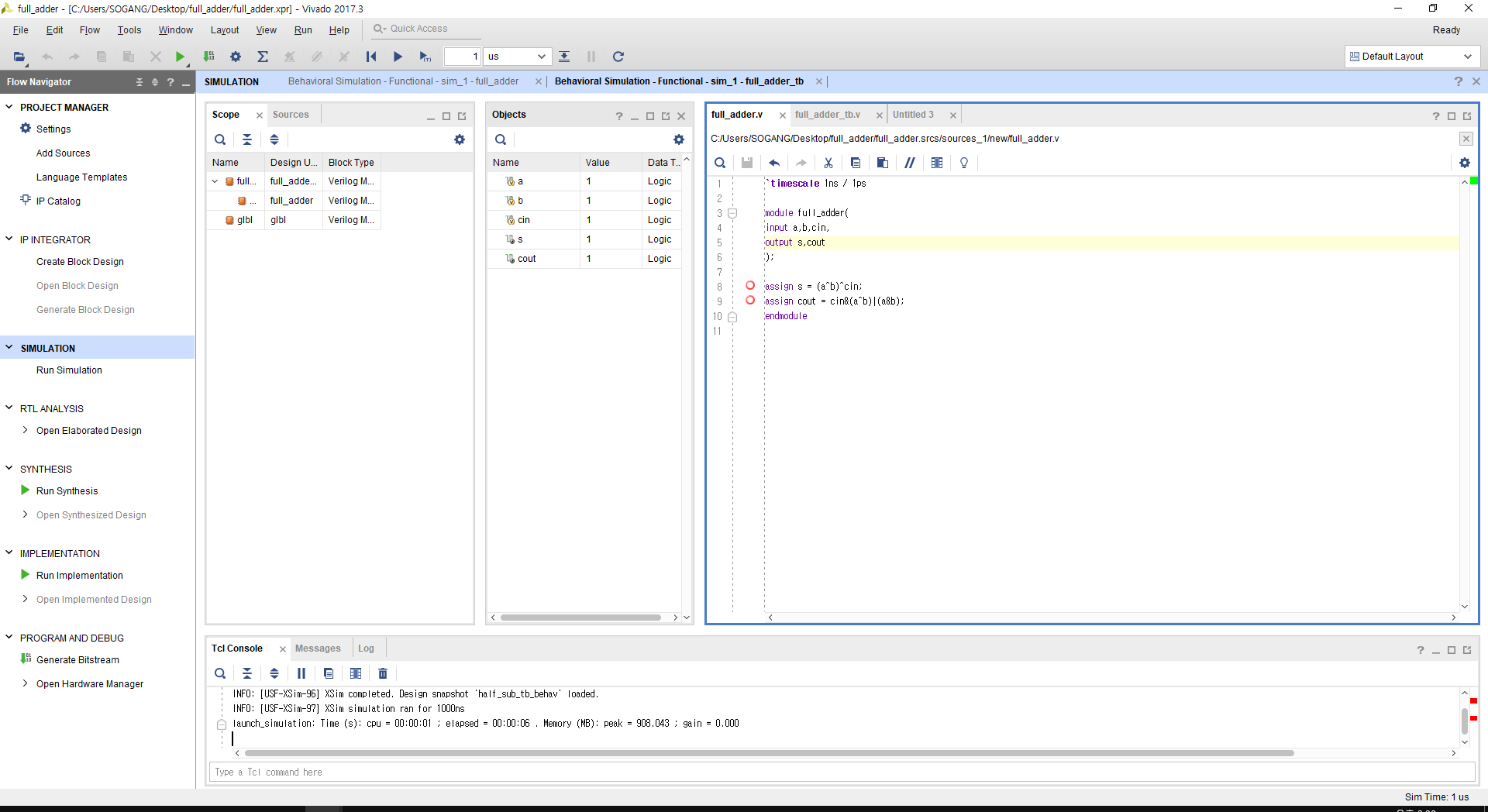
이론의 Boolean 함수와 일치한다.

Full Adder는 이전 자릿수의 자리올림인 C-in 또한 더할 수 있는 가산기이다. 따라서 3가지의 input인 a, b, cin 값을 가지고, Half-adder 처럼 두 가지의 output을 가진다. (Sum, Carry-out)

도표, 스케치, 그림, 라인이(가) 표시된 사진

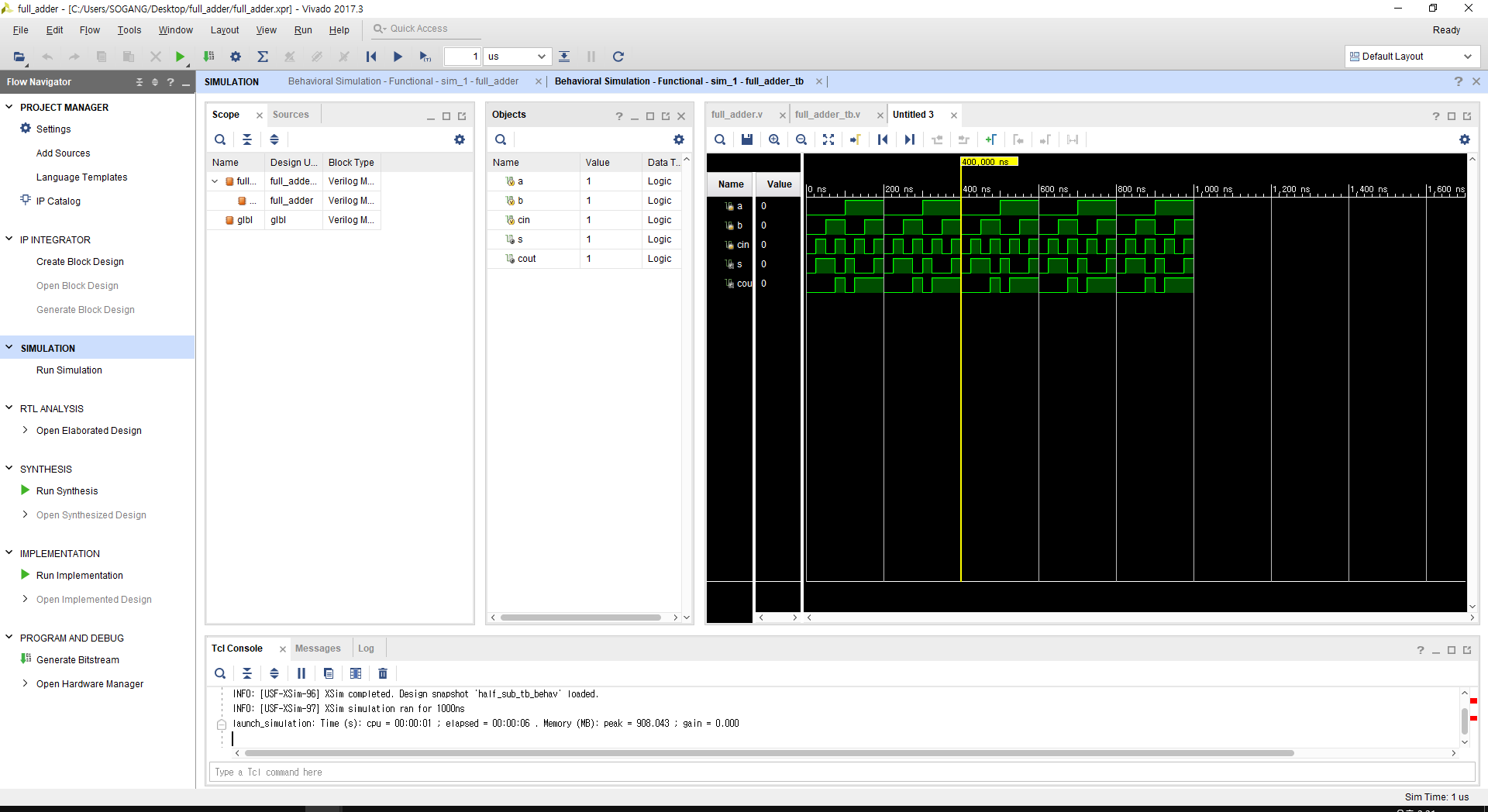
자동 생성된 설명

* Full Adder의 구조를 (a)로 Verilog 코딩한 결과



S의 경우에는 a와 b의 XOR 연산 후, 또 다른 input 값인 Cin 값과의 XOR 연산을 통해서 출력할 수 있고, Cout(Carry-out)의 경우에는 a와 b의 XOR 연산 후 cin 값과 AND 연산 후 a&b의 값으로 OR 연산을 취해줬다.

* Verilog의 simulation 결과와 진리표



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | b | Cin | S | Cout | (a ⊕b) ⊕Cin | Cin(a ⊕b) + ab |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

이론의 식과 동일하다.

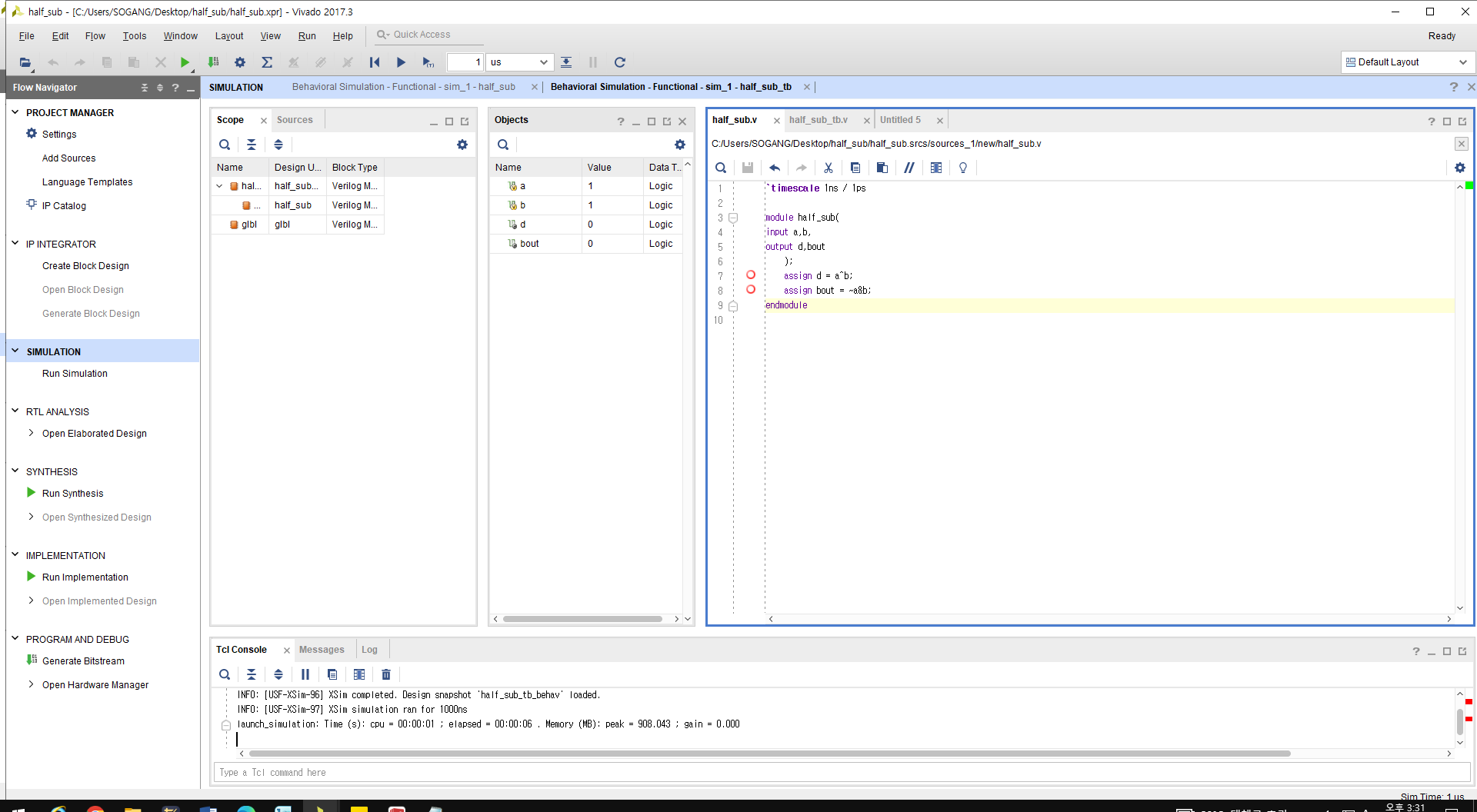
3.Full subtractor 및 Half substractor의 simulation 결과 및 과정에 대해서 설명하시오.

Half Subtractor는 1-bit 수의 뺄셈을 만드는데, 2개의 1 bit 입력, A,B이 존재하고, 두 값의 차이인 Difference(D)와 자리빌림 값인 Borrow(B), 두 개의 출력 값을 가진다.

도표, 라인, 스케치, 기술 도면이(가) 표시된 사진

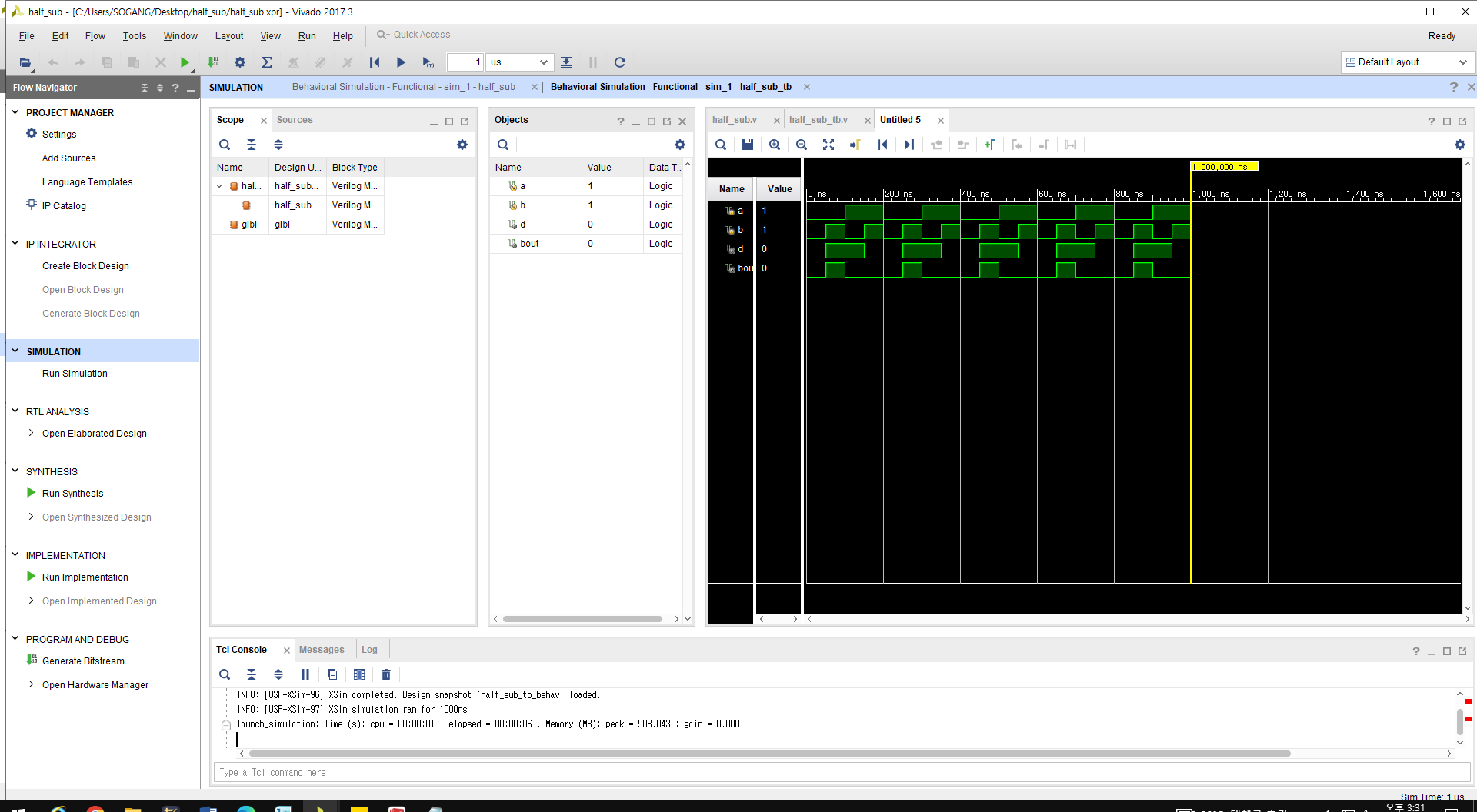
자동 생성된 설명

* Half Subtractor의 구조를 (a)로 Verilog 코딩한 결과



Half Subtractor의 D 값의 경우는 input a. b일 때 a와 b의 XOR 연산으로 구현 가능하다. 또한 자리빌림 값인 Bout의 경우에는 a’b의 값으로 구현 가능하다.

* Verilog의 simulation 결과와 진리표



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | D | B | a⊕b | a’&b |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |

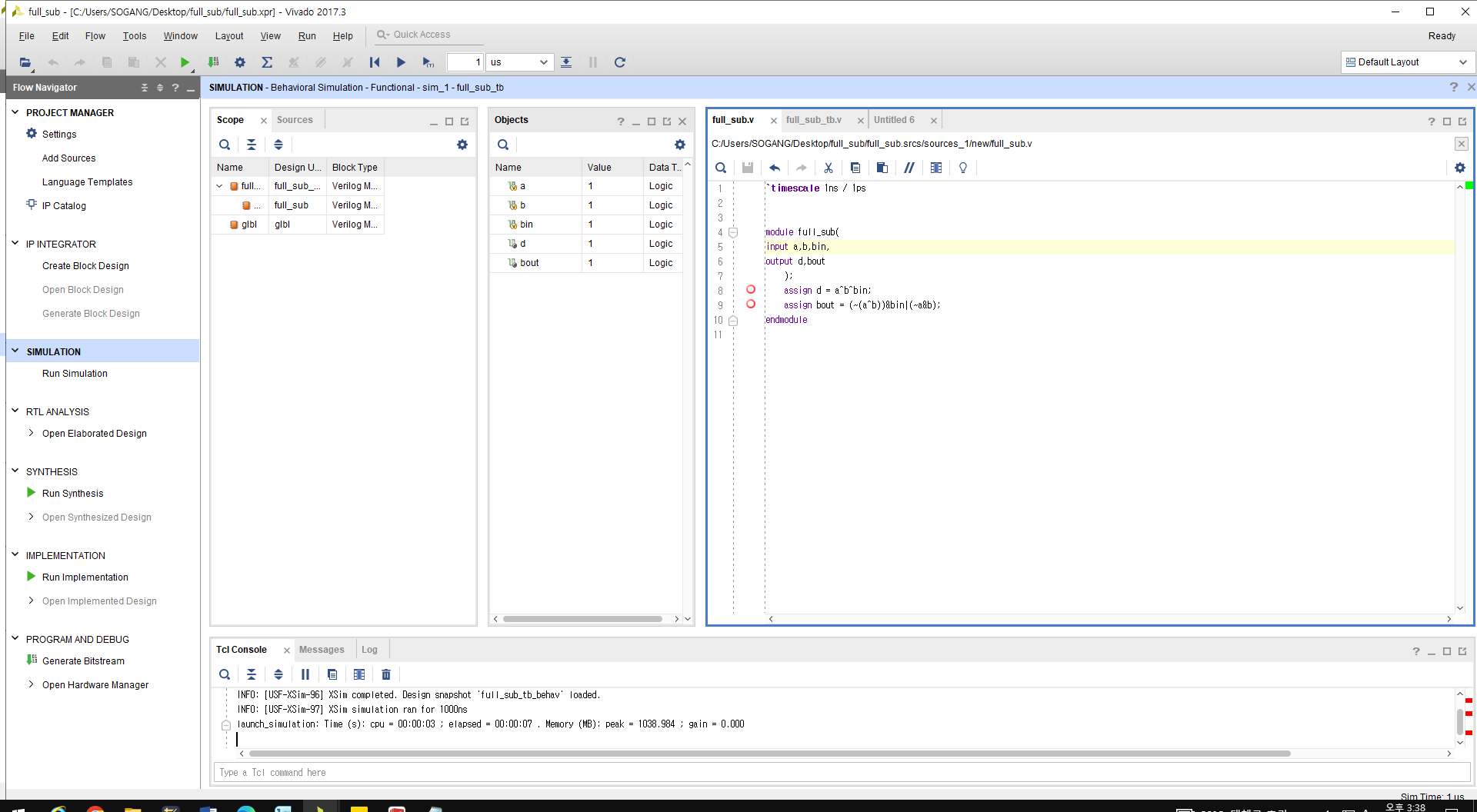
이론의 논리식과 동일하다.

Full-subractor의 경우에는 이전 Borrow(자리 빌림)값도 입력이 되어, 입력값이 a, b, borrow(in) 값이고, 출력값으로 Difference와 Borrow값 모두 출력한다. 출력값으로 나오는 Difference 값은 a⊕b⊕borrow(in) 값으로 나타낼 수 있고, Borrow(out)값은 (a⊕b)’bout + a’b라고 할 수 있다.

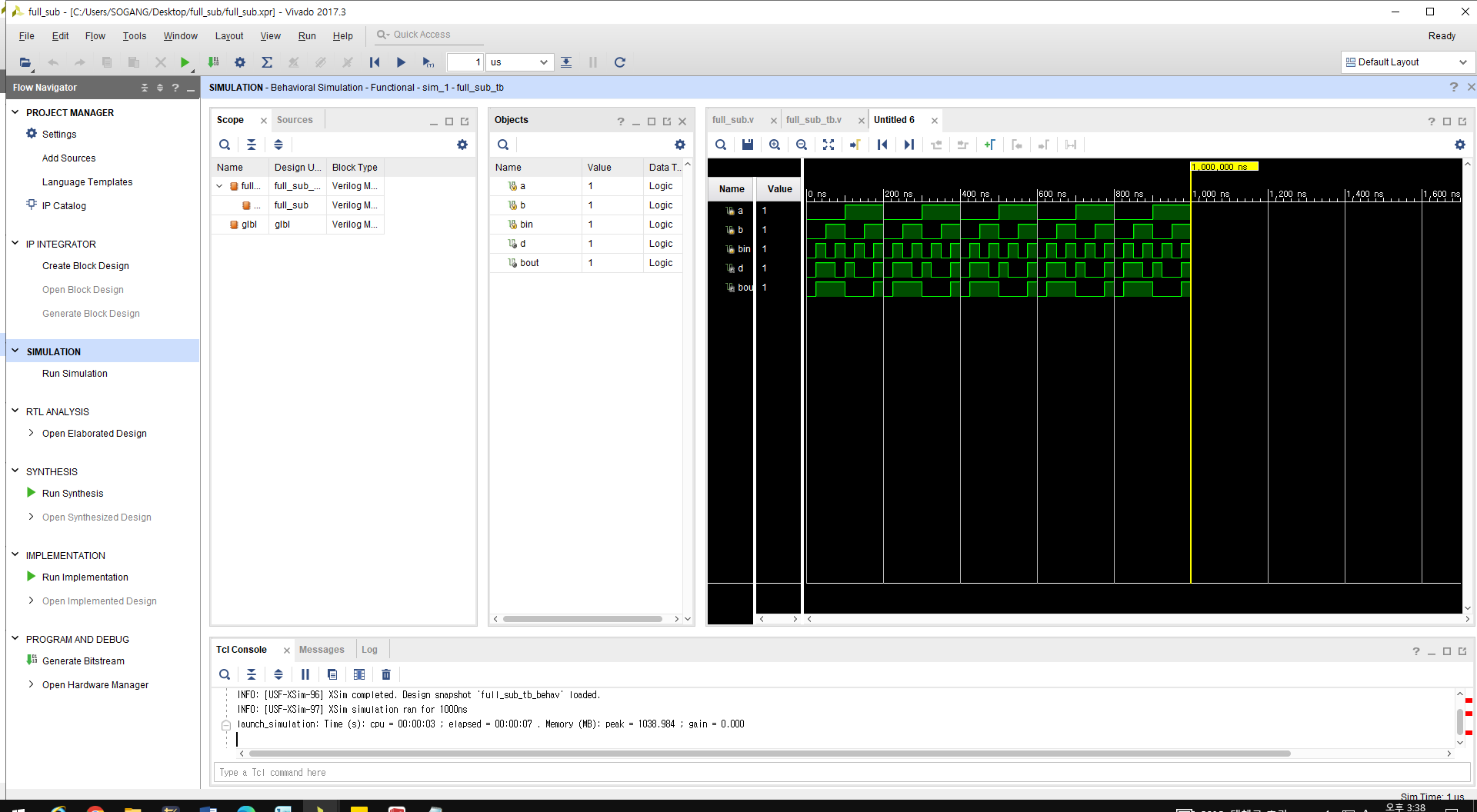
도표, 스케치, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

* Full Subtractor의 구조를 (a)로 Verilog 코딩한 결과



* Verilog의 simulation 결과와 진리표



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | b | bin | D | Bout | (a ⊕b) ⊕bin | ~(a⊕b)&bout| ~a&b |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

4. 8421(BCD)-2421 Code converter simulation 결과 및 과정에 대해서 설명하시오.

(진리표 작성 및 카르노맵 SOP form, POS form 포함)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8421 code | | | | 2421 code | | | |
| 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 |

위 진리표를 통해서 카르노 맵을 구성할 수 있다.

8421-2421을 할 때 input a, b, c, d는 8421 code 형식으로 입력되고, output x, y, z, w는 2421 code 형식으로 출력되어야 한다. 따라서 카르노 맵을 그려보면

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | AB | | | |
| 00 | 01 | 11 | 10 |
| CD | 00 | 0 | 0 | X | 1 |
| 01 | 0 | 1 | X | 1 |
| 10 | 0 | 1 | X | X |
| 11 | 0 | 1 | X | X |

X라고 표기한 부분은 1과 0 모두 바꿀 수 있다. 하지만, SOP 형식을 위해선 1을 기준으로 잡아야 하기 때문에 W = a + bc + bd가 나오게 된다. 숫자 0을 기준으로 했을 때는 POS 형식인 (a + b)(a + c + d)가 된다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | AB | | | |
| 00 | 01 | 11 | 10 |
| CD | 00 | 0 | 1 | X | 1 |
| 01 | 0 | 0 | X | 1 |
| 10 | 0 | 1 | X | X |
| 11 | 0 | 1 | X | X |

X의 경우에, SOP형식으로 a + bc + bd’가 된다. POS 형식은 (a+b)(a + c + d’)이다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | AB | | | |
| 00 | 01 | 11 | 10 |
| CD | 00 | 0 | 0 | X | 1 |
| 01 | 0 | 1 | X | 1 |
| 10 | 1 | 0 | X | X |
| 11 | 1 | 0 | X | X |

Y는 SOP 형식으로 a + b’c + bc’d, POS 형식으로는 (a + b + c)(a + c + d)(b’ + c’)이다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | AB | | | |
| 00 | 01 | 11 | 10 |
| CD | 00 | 0 | 0 | X | 0 |
| 01 | 1 | 1 | X | 1 |
| 10 | 1 | 1 | X | X |
| 11 | 0 | 0 | X | X |

Z는 SOP과 POS 형식으로 모두 d’에 해당한다.

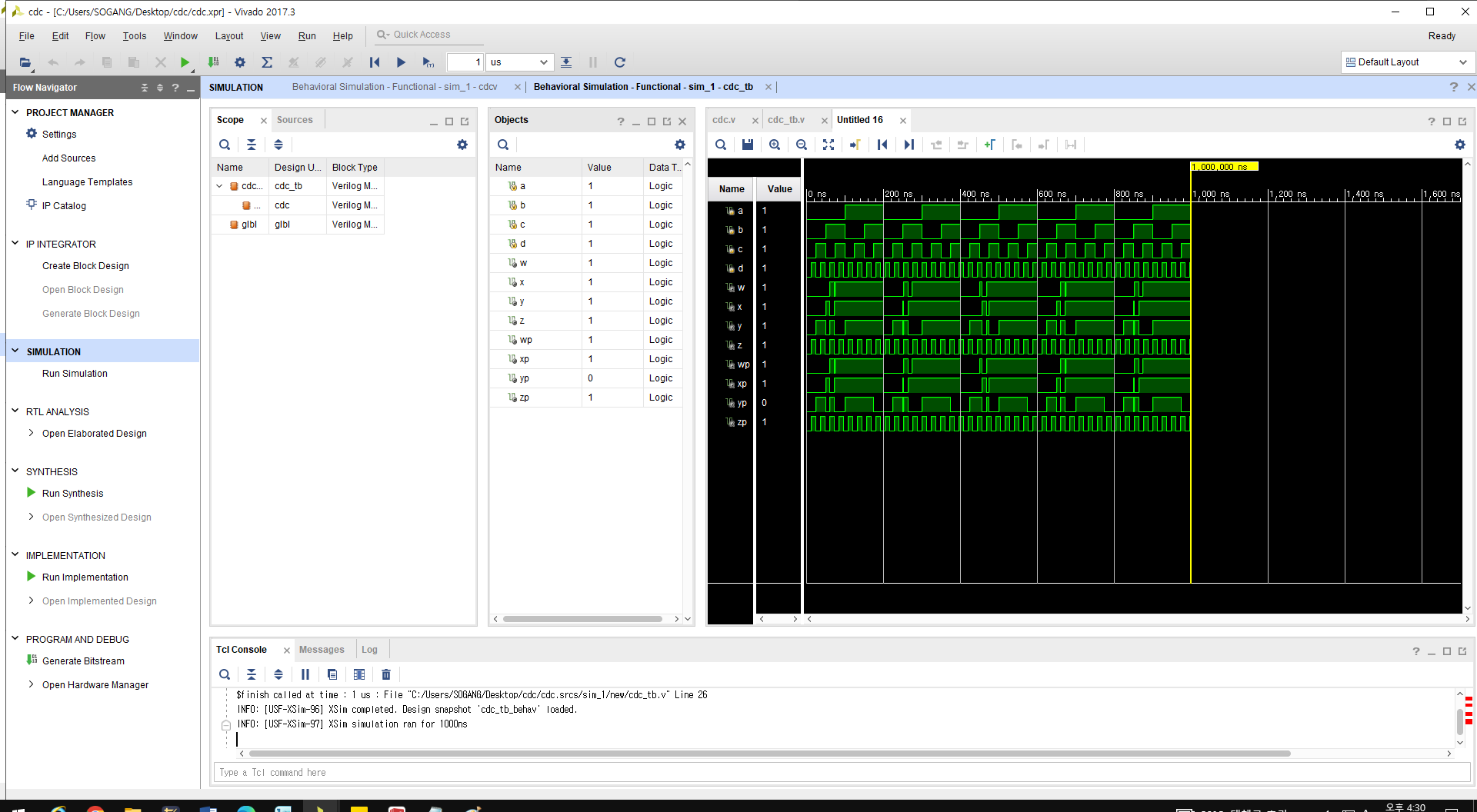
따라서 정리하여, Verilog 코딩을 하면 아래와 같다. Output x, y, z, w,는 SOP 형식으로, 테, yp, zp, wp는 모두 POS 형식으로 코딩했다.

* Verilog 코딩한 결과

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

- Simulation 결과

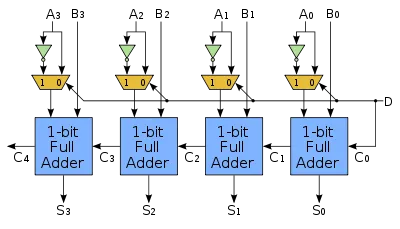


5. 결과 검토 및 논의사항

이번 실습 때 진행했던 전가산기 및 전감산기를 통해서 이전 자리의 올림 수와 빌림 수를 계산하는 것을 확인하여, 여러 bit의 수 +, - 연산이 가능함을 확인했다. 또한, 8421-2421 converter를 코딩하며 카르노 맵을 응용 학습하며, 2의 거듭제곱의 개수로 이웃하는 항들을 묶는 과정을 연습하고 논리회로를 설계할 수 있었다.

6. 추가 이론 조사 및 작성

실습에서 진행했던 가산기를 제외하고, 여러 가산기가 있는데, Arithmetic Logic Unit을 설명하겠다. 2의 보수법을 사용하여 뺄셈 연산을 구현한다. 이 ALU의 구성은 덧셈을 위한 가산기를 중심으로 구성되었다. 또한, 컴퓨터의 뺄셈을 구현하기 위해서는 2의 보수법을 적용해야 하기 때문에, 뺄셈의 수를 보수로 바꾸어 가산기에 입력하도록 하여 덧셈을 하도록 한다. 또한 덧셈의 과정을 반복하여 곱셈을, 뺄셈의 과정을 반복하여 나눗셈을 구현할 수 있다. 이러한 산술연산과 또한 XOR, AND, OR 과 같은 논리연산을 계산할 수 있다.



* 추가로 이 모든 실습에 사용되었다, testbench 파일을 작성하겠다. 순서대로 half-adder, full-adder, half-subtractor, full-subtractor, code converter이다.

