6주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231561 이름: 심소현

**1.**

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

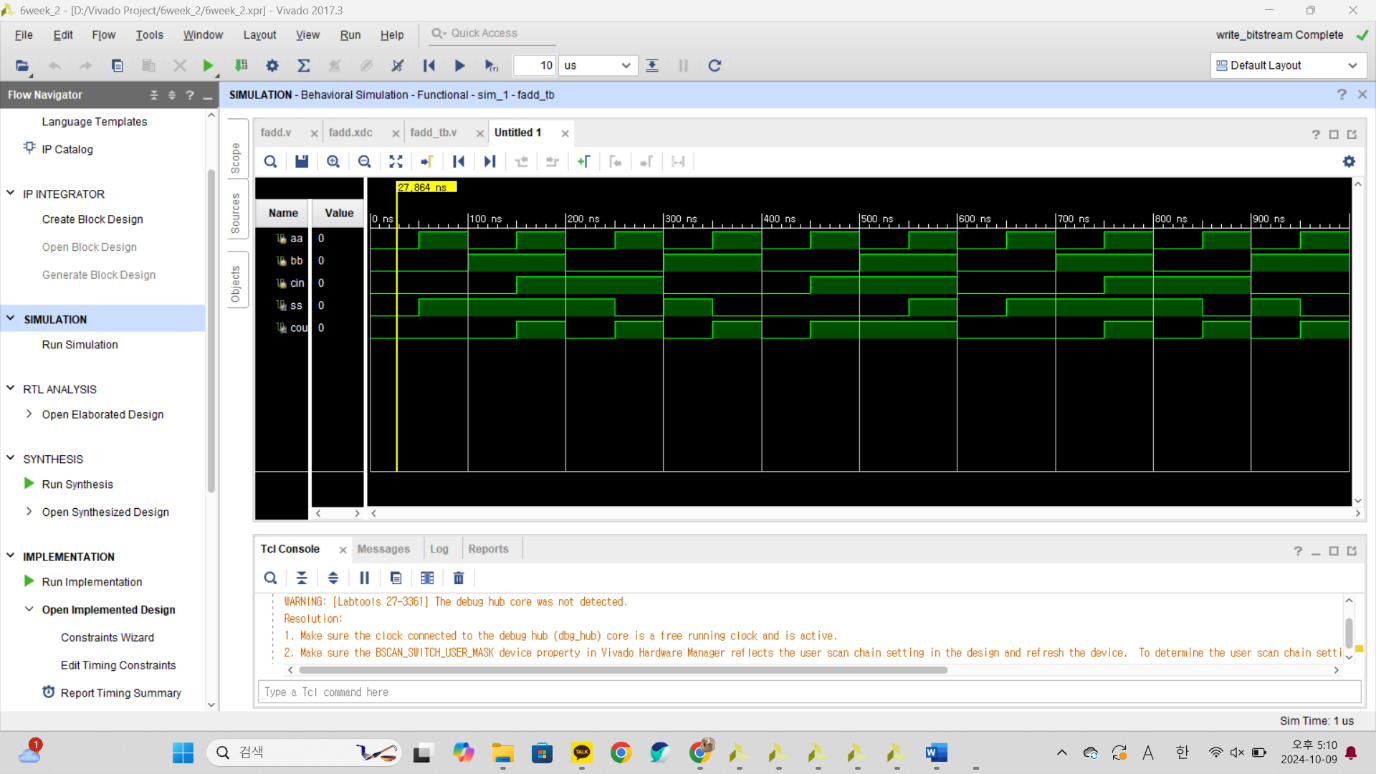
이번 주차 실험의 목적은 Adder와 Subtractor가 무엇인지 알아보고 Half의 경우와 Full의 경우에 따라 각자 확인하는 것이다. 또한 Code Converter를 이해하기 위하여 변환하는 과정을 카르노 맵으로 작성해보고 이를 바탕으로 식을 정리하여 직접 Verilog로 구현하려 하였다. Adder, Subtractor, Code Converter를 모두 &, |, ^로 구현하며 과정을 확인할 수 있었다.

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

**2.**

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

아래는 Full Adder 연산 회로를 Verilog로 구현한 것의 simulation이다.

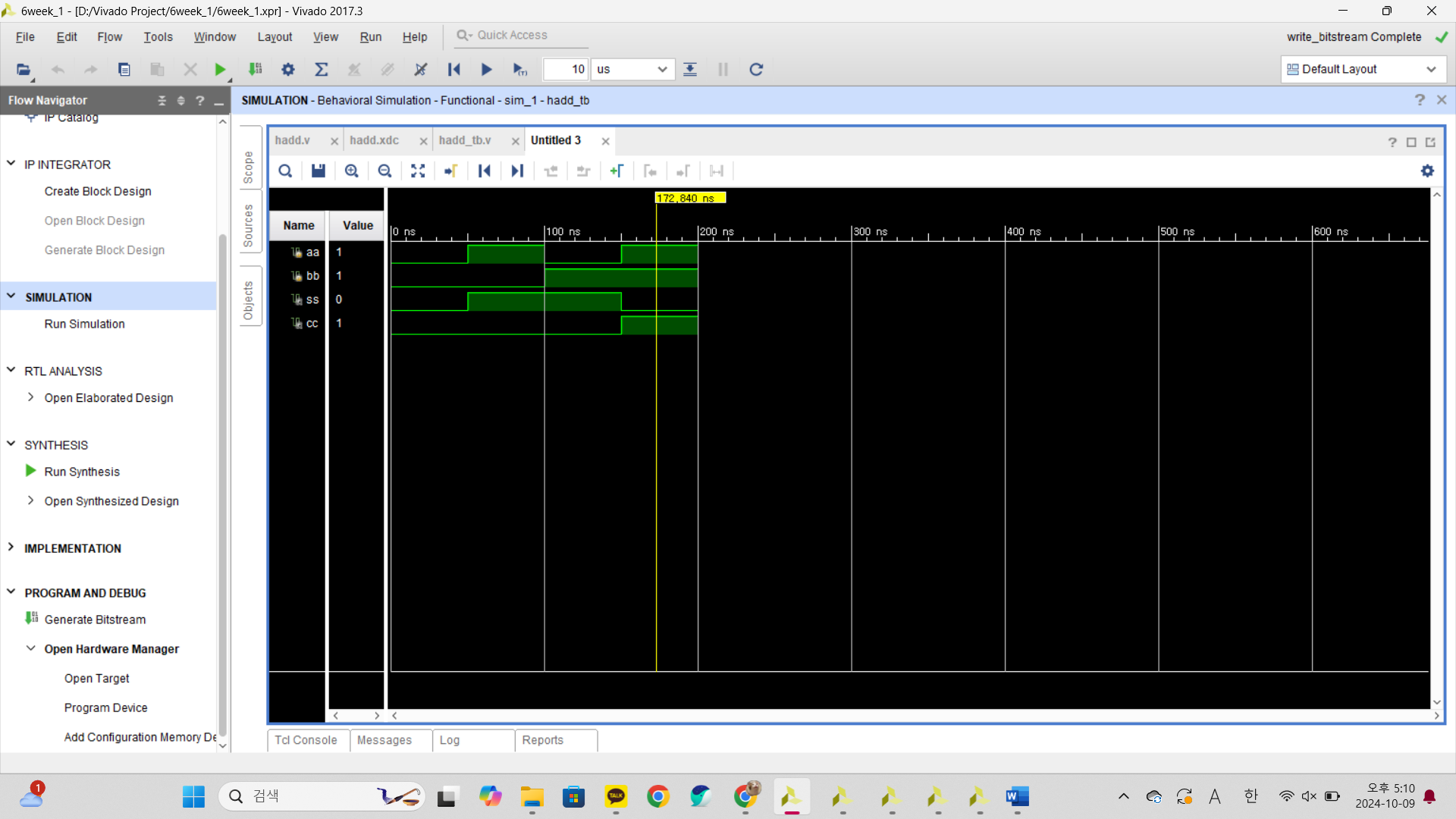


아래는 Full Adder 연산 회로의 진리표이다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| C | B | A | S | C |
| 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 |

Full Adder 연산 회로는 3개의 입력을 받아 2개의 출력을 내는 구조이다. 우선 입력은 A, B라는 2개의 기본 입력와 Carry 입력(Cin)으로 구성되어 있다. 출력으로는 우선 A와 B가 XOR 게이트를 거쳐서 이루어진 출력이 Cin과 XOR 게이트를 거쳐 구성되는 S가 있다. 또한, A와 B가 XOR 게이트를 거쳐 이루어진 출력과 Cin이 AND 게이트를 거친 출력물이 A와 B가 AND 게이트를 거쳐 만들어진 출력물과 OR 게이트를 거쳐서 완성되는 Cout 출력이 있다. Full Adder는 자리올림수 Carry를 받아들이지 못하는 Half Adder와는 달리 여러 자리의 이진수 덧셈을 가능하게 해준다.

아래는 Half Adder 연산 회로를 Verilog로 구현한 것의 simulation이다.



아래는 Half Adder 연산 회로의 진리표이다.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| A | B | S | C |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

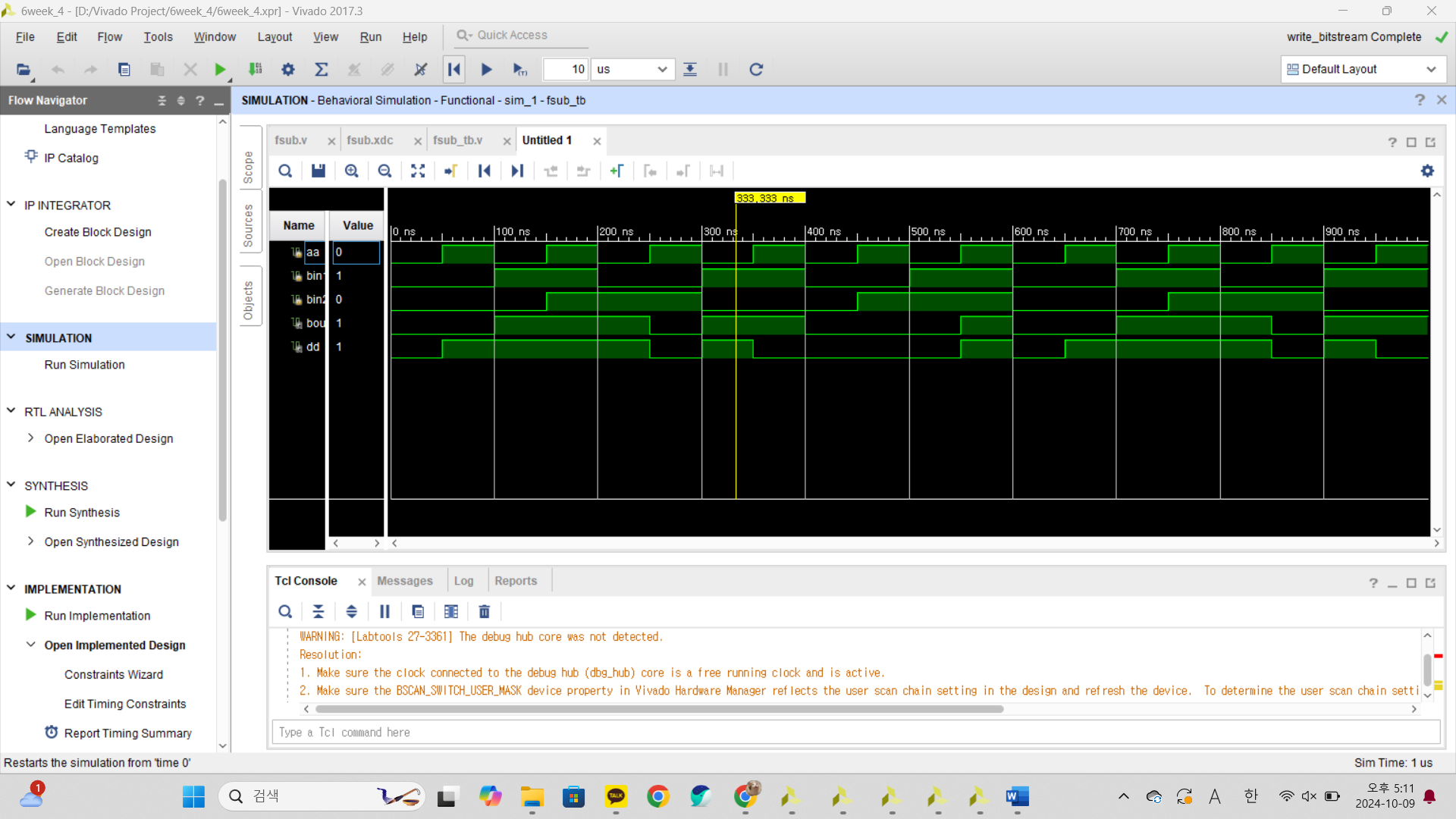
Half Adder 연산 회로는 2개의 입력을 받아 2개의 출력을 내는 구조이다. 우선 입력은 A, B라는 2개의 기본 입력으로 구성되어 있다. 출력으로는 우선 A와 B가 XOR 게이트를 거쳐서 이루어진 출력인 S가 있다. 또한, A와 B가 AND 게이트를 거친 출력물 C가 있다. Half Adder는 두 개의 한 자릿수 이진수 덧셈만 가능하며 출력 C가 이때 발생하는 자리올림수를 뜻한다. 하지만 입력에서는 자리올림수를 받을 수 없기 때문에 Carry의 입력이 필요한 경우에는 Full Adder를 사용해야 한다.

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

**3.**

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

아래는 Full Subtractor 연산 회로를 Verilog로 구현한 것의 simulation이다.

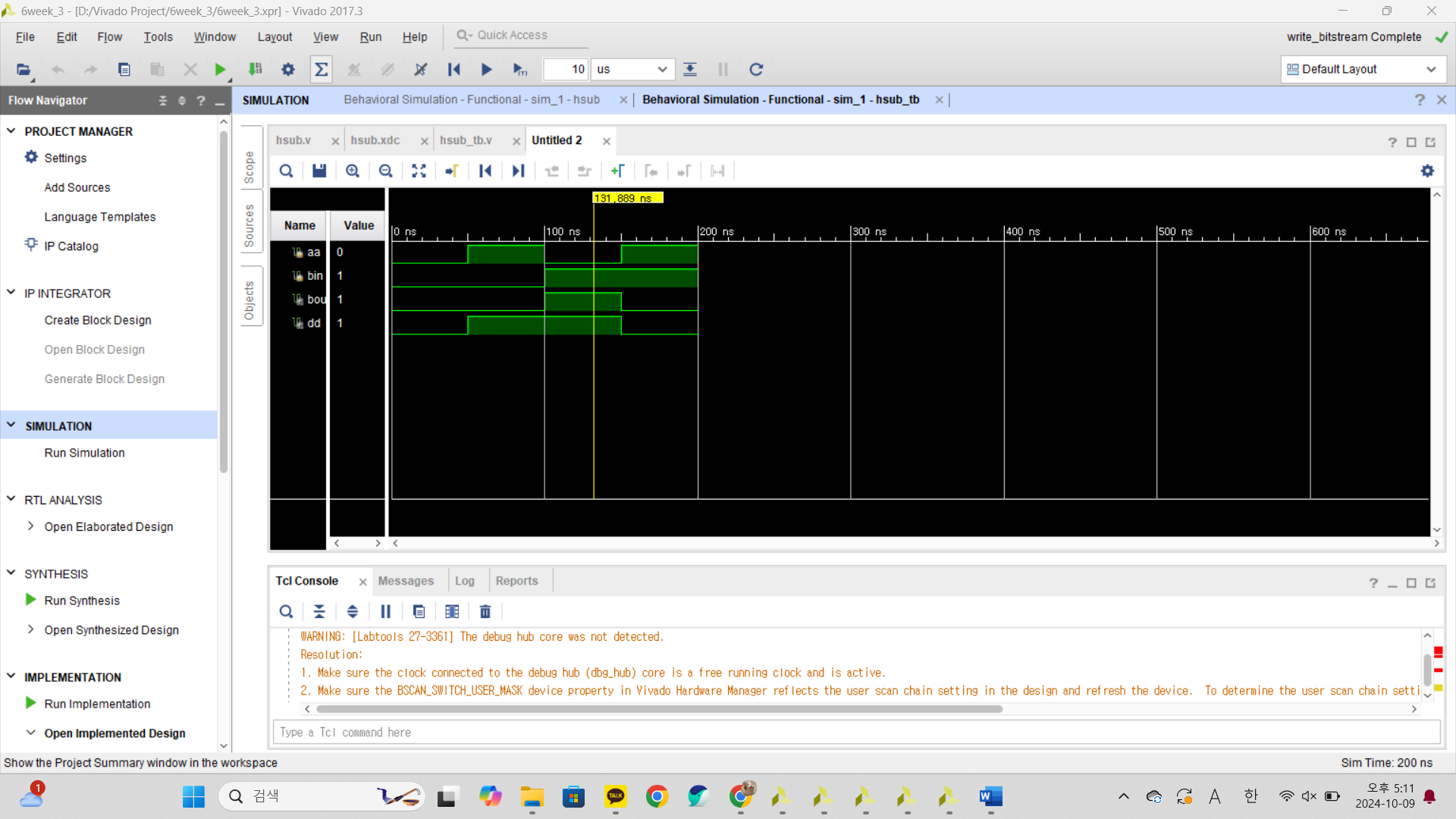


아래는 Full Subtractor 연산 회로의 진리표이다.

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

Full Subtractor 연산 회로는 3개의 입력을 받아 2개의 출력을 내는 구조이다. 우선 입력은 A, Bn라는 2개의 기본 입력와 Borrow 입력(Bn-1)으로 구성되어 있다. 출력으로는 우선 A와 B가 XOR 게이트를 거쳐서 이루어진 출력이 Bn-1와 XOR 게이트를 거쳐 구성되는 D가 있다. 또한, A와 B가 XOR 게이트를 거쳐 이루어진 출력이 NOT 게이트를 거친 값과 Bn-1이 AND 게이트를 거친 출력물이 A와 B’가 AND 게이트를 거쳐 만들어진 출력물이 OR 게이트를 거쳐서 완성되는 bn 출력이 있다. Full Subtractor는 빌림수를 받아들이지 못하는 Half Subtractor와는 달리 빌림수(Bn-1)까지 입력받는다.

아래는 Half Subtractor 연산 회로를 Verilog로 구현한 것의 simulation이다.



아래는 Half Subtractor 연산 회로의 진리표이다.

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

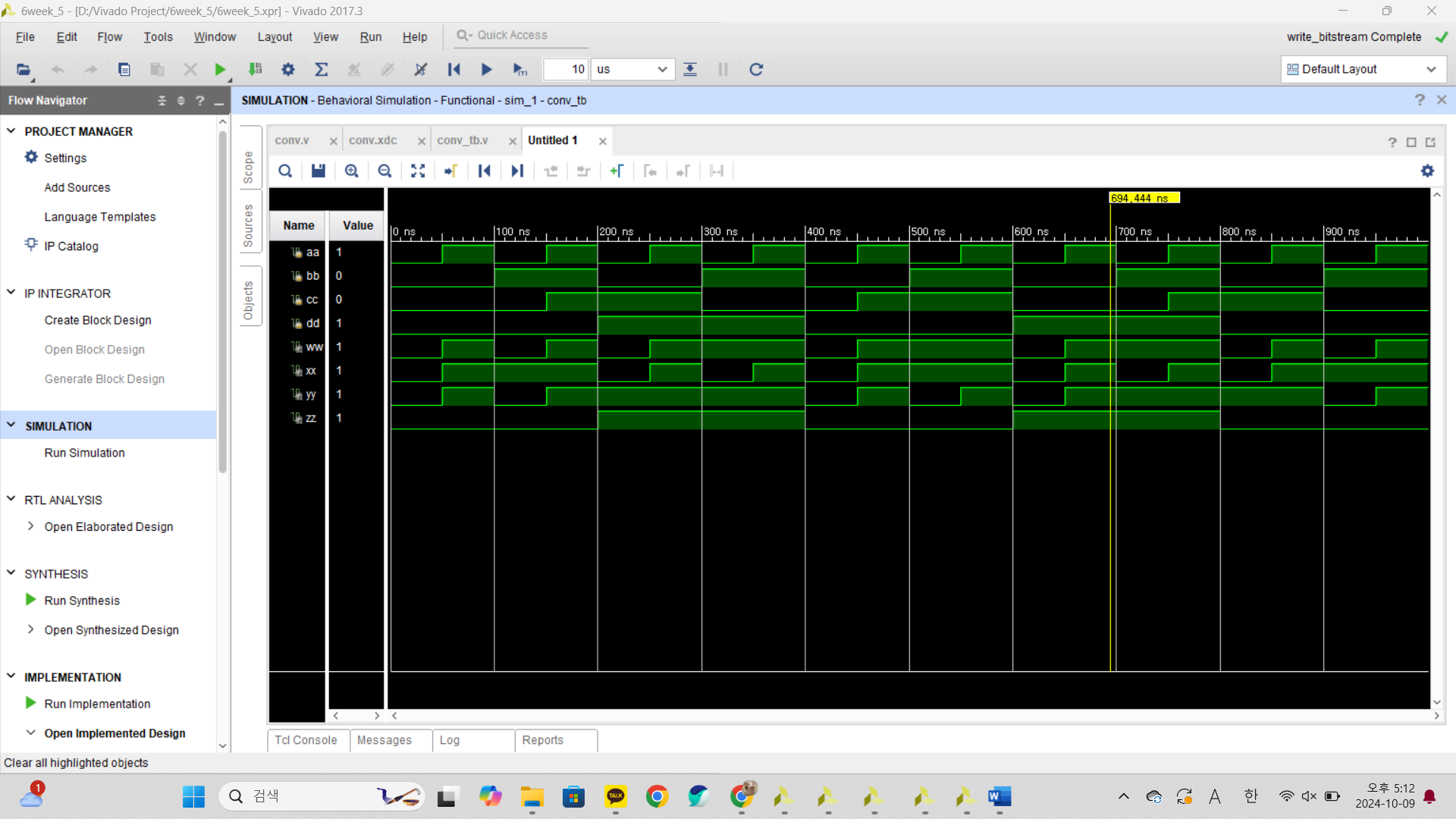
Half Subtrator 연산 회로는 2개의 입력을 받아 2개의 출력을 내는 구조이다. 우선 입력은 A, B라는 2개의 기본 입력으로 구성되어 있다. 출력으로는 우선 A와 B가 XOR 게이트를 거쳐서 이루어진 출력인 ㅇ가 있다. 또한, A와 B’가 AND 게이트를 거친 출력물 b가 있다. Half Adder는 빌림수의 출력은 b로 존재하지만 빌림수를 입력 받을 수는 없는 구조이다.

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

**4.**

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

아래는  8421(BCD)-2421 Code converter를 Verilog로 구현한 프로그램의 simulation이다.



아래는 8421(BCD)-2421 Code converter의 진리표이다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8424 BCD | | | | 2421 BCD | | | |
| A | B | C | D | W | X | Y | Z |
| 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 |

아래는 8421(BCD)-2421 Code converter의 카르노 맵이다.

W

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

SOP 식은 W = A + BC + BD이다. POS 식은 W = (A+B)\*(A+C’)\*(A+D)이다.

X

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

SOP 식은 X = A + BD’ + BC이다. POS 식은 X = (A+B’)(A+D)(A+C)이다.

Y

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

SOP 식은 Y = A + B’C + BC’D이다. POS 식은 Y = (A+B)(A+C’)(A+D)이다.

Z

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

SOP 식은 Z = D이다. POS식 또한 Z = D이다.

8421(BCD)-2421 Code converter에서는 우선 4비트의 BCD 입력을 받는다. 그 후 각각의 자릿수에 대한 게이트를 거쳐 2421 Code로 변환되어 출력된다. 이때 게이트의 구성은 위의 SOP식을 논리 회로로 구성한 것과 동일하다.

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

**5.**

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

결과를 보았을 때, simulation 결과가 진리표와 일치하는 것을 확인할 수 있었다. 연산 회로를 구성할 때는 Half의 경우보다 Full의 경우가 더 복잡한 계산을 요구하기 때문에 각각의 출력을 구현하기 위하여 더 많은 게이트를 사용해야 한다. 간단한 한 자릿수 계산에서는 Half 연산을 사용해도 문제 없지만 많은 자릿수에서는 Full을 사용해야 오류 없이 계산을 진행할 수 있다. 8421(BCD)-2421 Code converter에서는 카르노 맵을 통해서 SOP식을 확인할 수 있었기에 카르노 맵의 편리성을 알 수 있었다.

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

**6.**

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

추가적으로 Adder와 Subtractor에 대하여 조사를 하였다. Adder, 즉 가산기는 데이터 처리 및 연산에서 필수적이다. 컴퓨터 연산 장치인 ALU, 데이터 전송 및 검증, 디지털 시계와 같은 경우에 사용한다. Subtractor, 즉 감산기는 가산기와 다르게 빼기 연산에서 사용한다. 산술 연산, 디지털 신호 비교와 같은 경우에서 주로 사용한다.

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