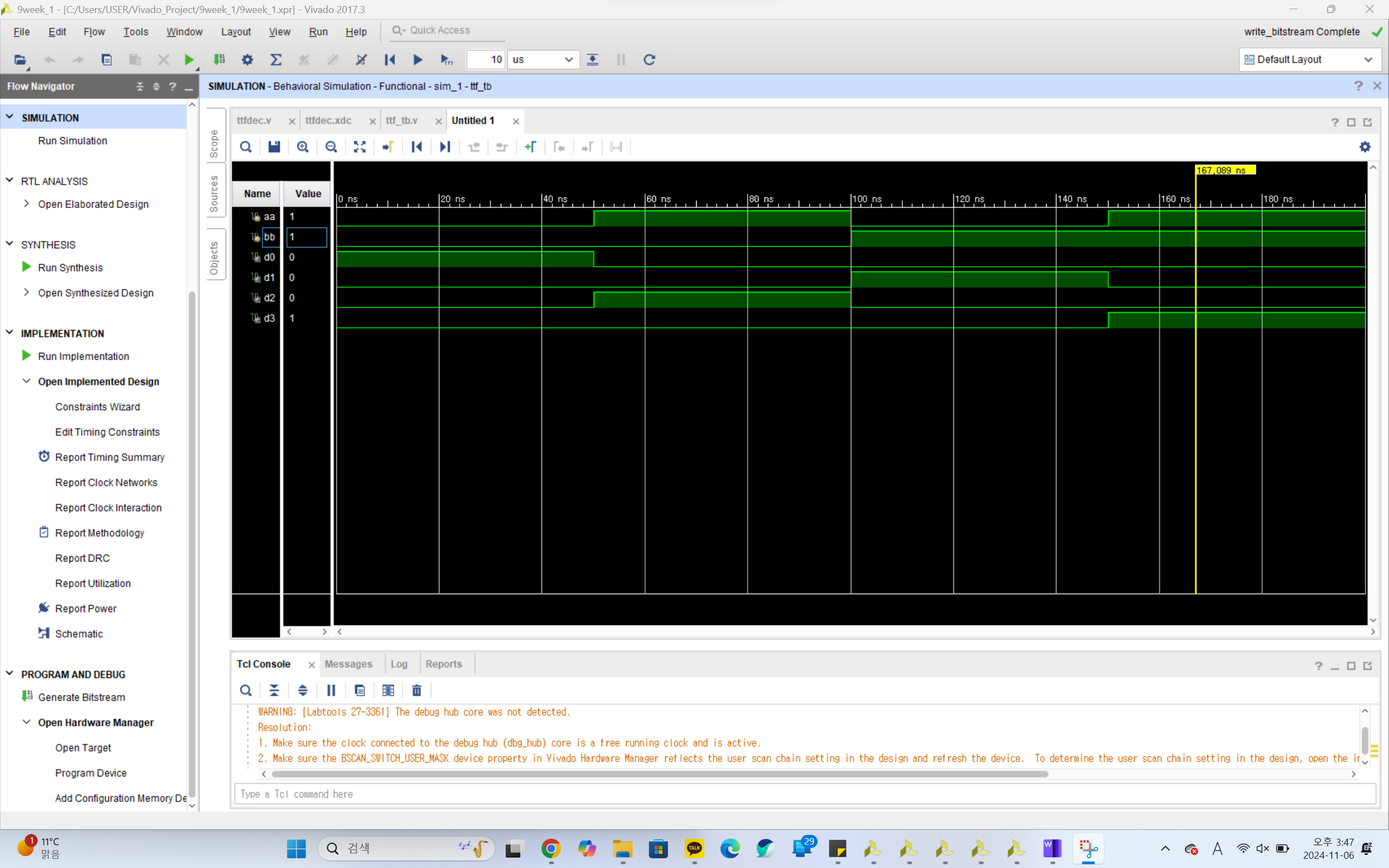
9주차 결과보고서

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

**1.**

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

2 to 4 Decoder에서 Active high의 simulation 결과는 다음과 같다.



2 to 4 Decoder에서 Active high의 truth table은 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | Output | | | |
| A | B | D0 | D1 | D2 | D3 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

2 to 4 Decoder에서 Active high의 k-map은 다음과 같다.

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |

d0

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |

d1

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 0 |

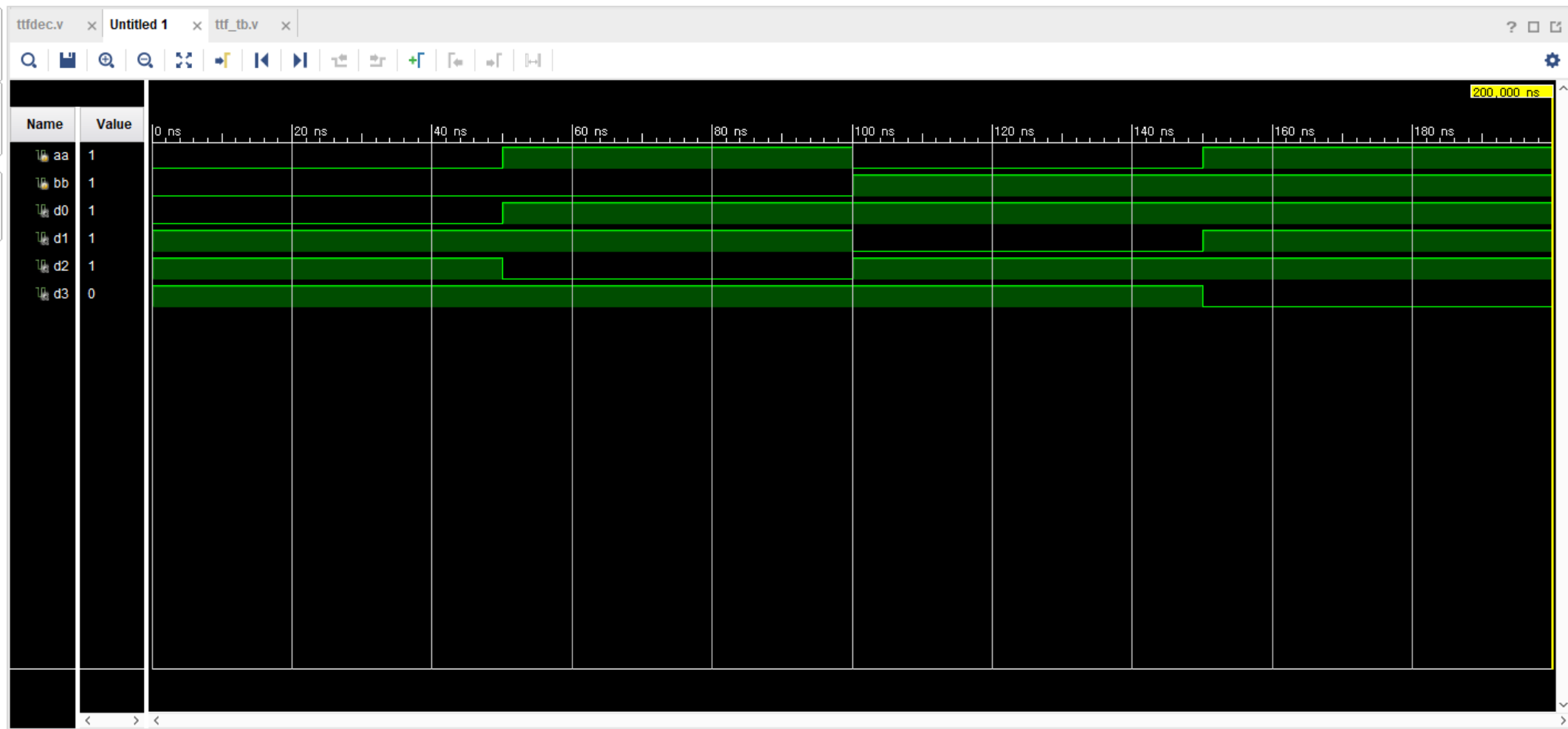
d2

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |

d3

2 to 4 Decoder는 2개의 입력 비트에서 4개의 출력 비트로 변환하는 작업을 수행한다. 그중 Active high는 선택된 출력이 1로 활성화되는 경우라고 볼 수 있다. 예를 들어, 입력이 00일 때 첫 번째 출력이 1이고 나머지는 다 0이 되는 것이라고 볼 수 있다.

2 to 4 Decoder에서 Active low의 simulation 결과는 다음과 같다.



2 to 4 Decoder에서 Active low의 truth table은 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | Output | | | |
| A | B | D0 | D1 | D2 | D3 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

2 to 4 Decoder에서 Active low의 k-map은 다음과 같다.

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 1 |

d0

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

d1

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |

d2

|  |  |  |
| --- | --- | --- |
| a b | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

d3

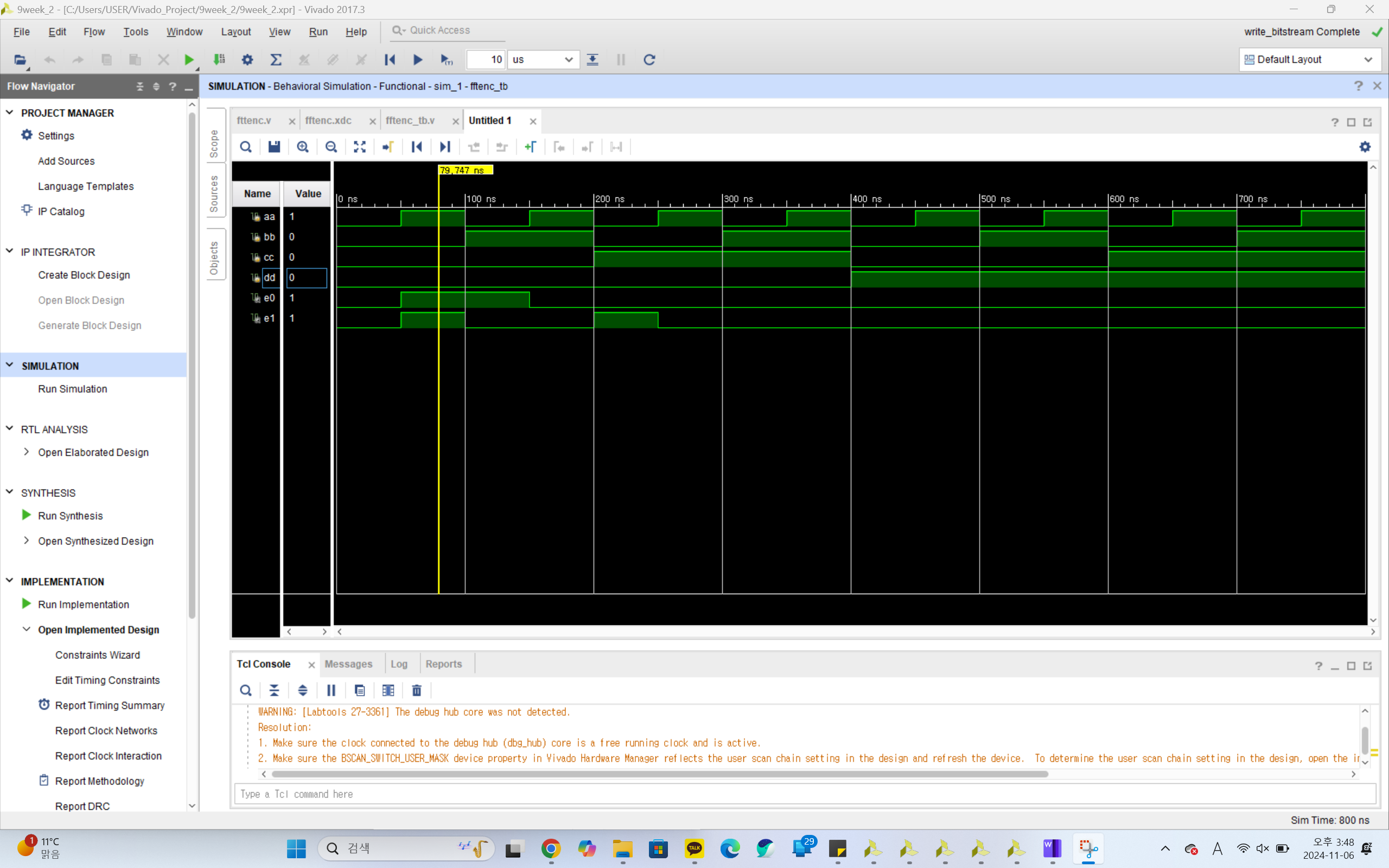
2 to 4 Decoder에서 Active low는 선택된 출력이 0으로 활성화되는 경우라고 볼 수 있다. 예를 들어, 입력이 00일 때 첫 번째 출력이 0이고 나머지는 다 1이 되는 것이라고 볼 수 있다.

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

**2.**

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

4 to 2 Encoder의 simulation 결과는 다음과 같다.



4 to 2 Encoder의 truth map은 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| input | | | | output | |
| A | B | C | D | E0 | E1 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |

4 to 2 Encoder의 k-map은 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab cd | 00 | 01 | 11 | 10 |
| 00 | X | 0 | X | 0 |
| 01 | 1 | X | X | X |
| 11 | X | X | X | X |
| 10 | 1 | X | X | X |

e0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab cd | 00 | 01 | 11 | 10 |
| 00 | X | 0 | X | 1 |
| 01 | 0 | X | X | X |
| 11 | X | X | X | X |
| 10 | 1 | X | X | X |

e1

4 to 2 Encoder는 4개의 입력 중 하나가 활성화 되면 그에 대응하는 2-bit의 출력을 생성하는 회로이다. 여러 개의 입력이 동시에 활성화된다면 우선순위가 필요하지만 위의 실습에서는 이를 상정하지 않았다.

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

**3.**

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

4 to 2 Encoder에서 입력 형태 4가지를 제외한 나머지 입력 형태는 여러 개의 입력이 동시에 나타나 기본적으로 4 to 2 Encoder의 4가지 입력만 고려한 회로에서는 제대로 처리되지 않는 경우이다. 이 때는 우선순위를 지정하여 어떤 출력을 활성화해야 하는지 나타내야 한다. 우선순위를 지정한 우선순위 인코더라면 4가지를 제외한 입력에서 어떤 입력을 우선적으로 선택해서 대응하는 결과를 출력한다.

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

**4.**

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

모든 입력 형태에 대응하여 동작되는 priority encoder는 다음 truth table을 따른다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| input | | | | output | |
| A | B | C | D | E0 | E1 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 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 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

이는 우선순위를 A, B, C, D의 순서로 지정한 priority Encoder이다. 이를 바탕으로 Verilog 코드를 구현하면 다음과 같다.

E0 = A | (B & ~A);

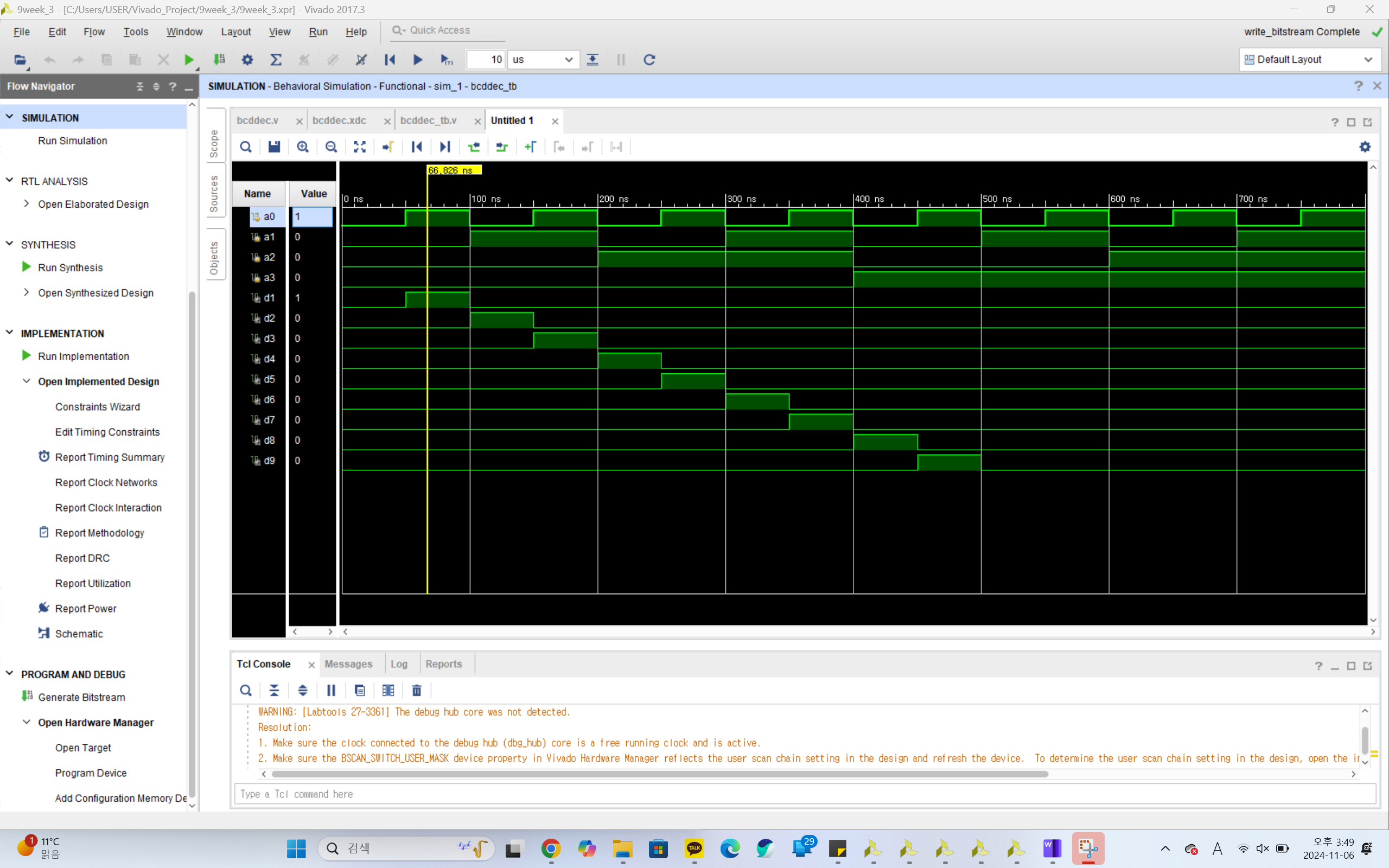
E1 = A | (C & ~B & ~A);

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

**5.**

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

BCD to Decimal decoder의 simulation 결과는 다음과 같다.



BCD to Decimal decoder의 truth table은 다음과 같다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BCD input | | | | Decimal output | | | | | | | | |
| A | B | C | D | d1 | d2 | d3 | d4 | d5 | d6 | d7 | d8 | d9 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

BCD to Decimal decoder의 k-map은 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3a2 a1a0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

d1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3a2 a1a0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

d2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3a2 a1a0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

d3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3a2 a1a0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

d4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3a2 a1a0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

d5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3a2 a1a0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

d6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3a2 a1a0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

d7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3a2 a1a0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 0 |

d8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a3a2 a1a0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 |

d9

BCD to Decimal Decoder는 4-bit의 입력을 받아 해당하는 10진 출력으로 변환하는 회로이다. 2진수의 값을 해당하는 10진수의 값으로 변환하여 이에 해당하는 위치의 출력을 1로 활성화한다. 그 외의 값들은 0으로 처리한다.

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

**6.**

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

Encoder의 응용에는 다음과 같은 예시들이 있다. 다중 입력 데이터의 크기를 줄이는 데이터 압축, 입력을 감지하고 CPU로 전달하는 등의 디지털 회로 사용, 우선순위 인코딩, 신호 전송 시 데이터를 인코딩하여 복구할 수 있도록 해주는 통신 시스템 등이 있다.

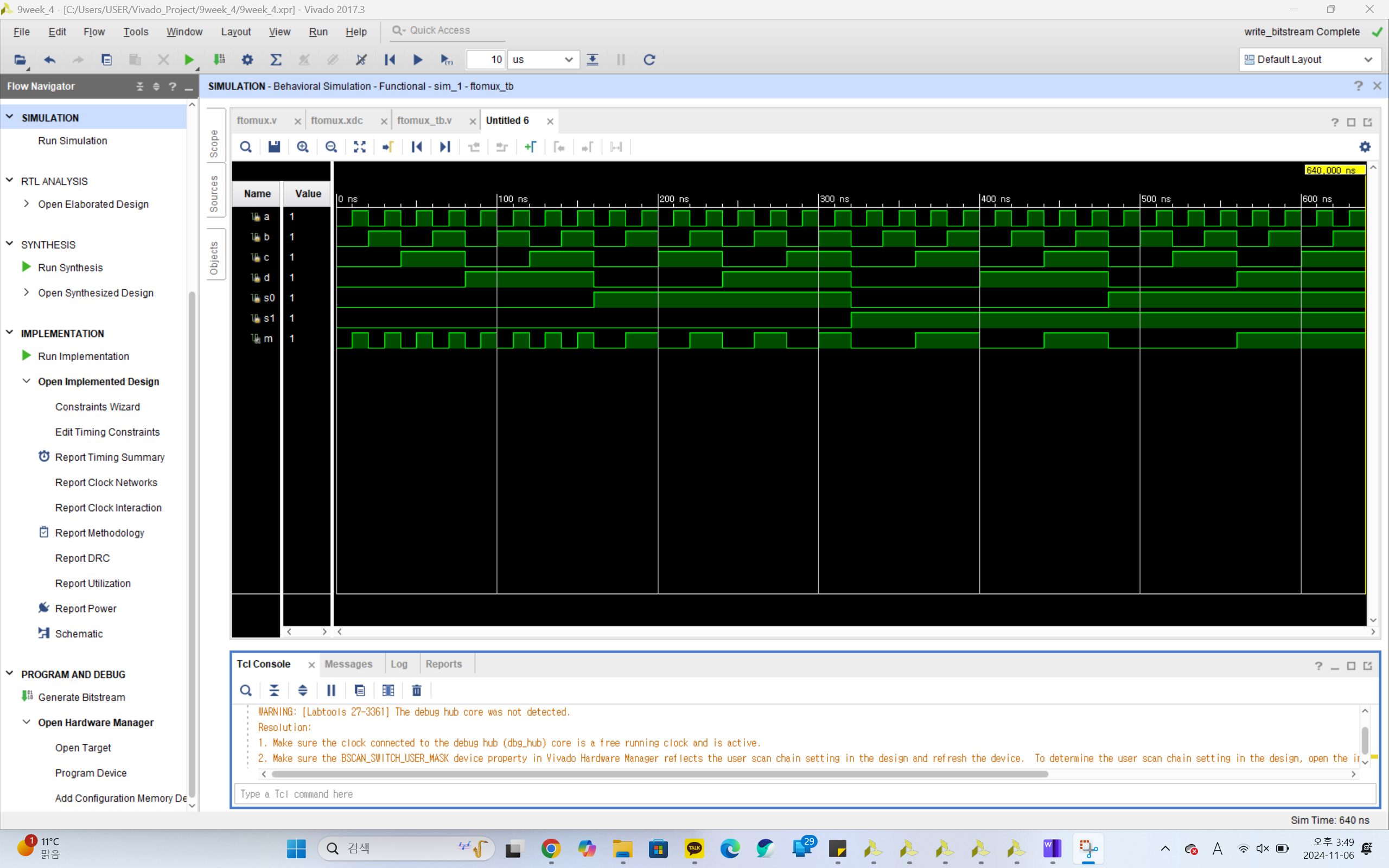
Decoder의 응용에는 다음과 같은 예시들이 있다. 비트를 변환하여 숫자 혹은 문자로 표현하는 것에 도움을 주는 디스플레이, 메모리 주소 해독, 인코딩 되어 전달된 데이터를 해독하는 통신 시스템, CPU 명령을 해석하는 프로세서, 등이 있다.

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

**7.**

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

4 to 1 line MUX의 simulation 결과는 다음과 같다.



4 to 1 line MUX의 truth table은 다음과 같다.

|  |  |  |
| --- | --- | --- |
| input | | output |
| s1 | s0 | m |
| 0 | 0 | input A |
| 0 | 1 | input B |
| 1 | 0 | input C |
| 1 | 1 | input D |

4 to 1 line MUX의 k-map은 다음과 같다.

s1s0 = 00

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab cd | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

s1s0 = 01

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab cd | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |

s1s0 = 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab cd | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 1 | 1 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 0 | 1 | 1 |
| 10 | 0 | 0 | 1 | 1 |

s1s0 = 11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ab cd | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 1 | 1 | 0 |

4 to 1 line MUX의 코드는 다음과 같다.

assign m = (a&~s0&~s1)|(b&s0&~s1)|(c&~s0&s1)|(d&s0&s1);

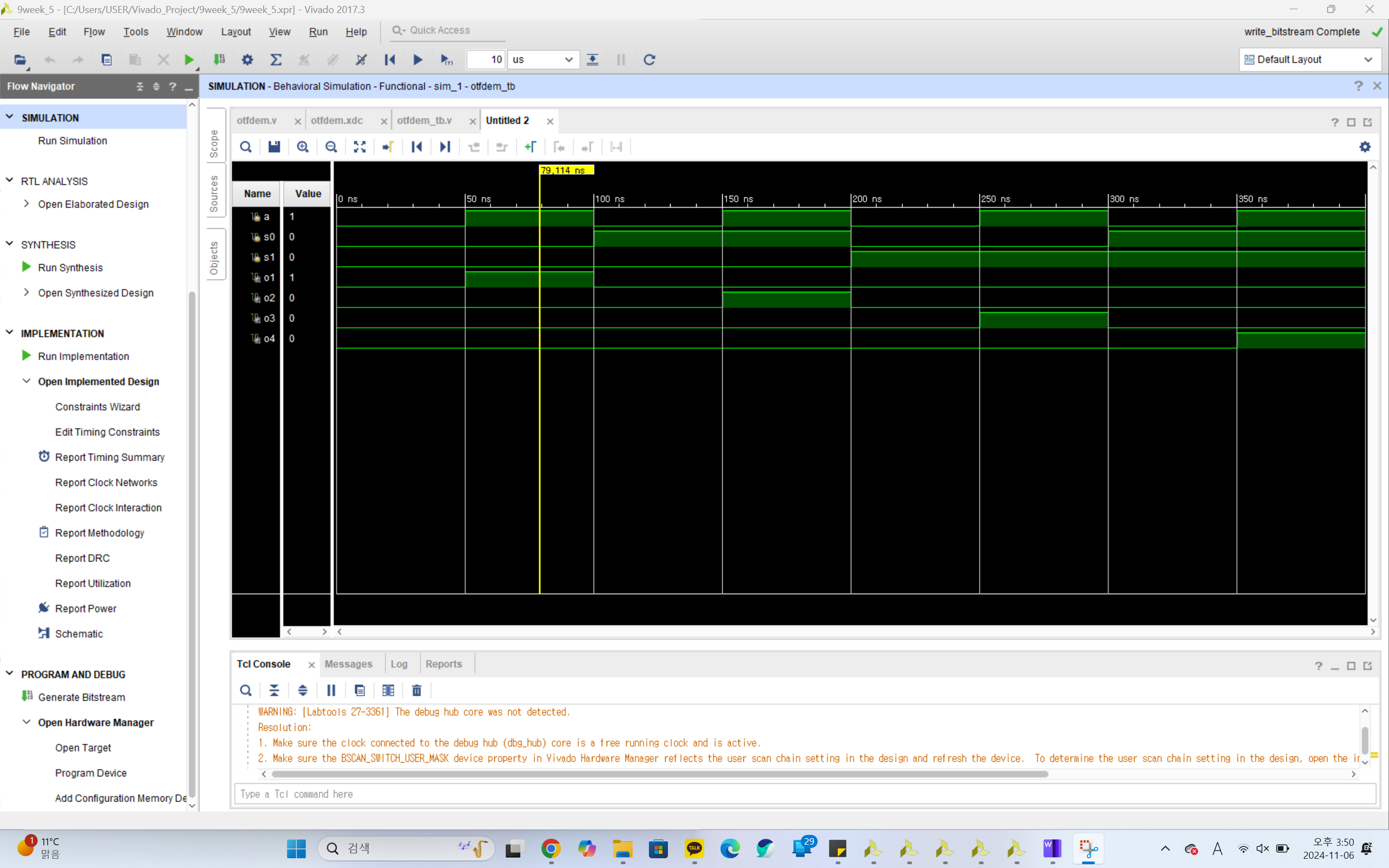
4 to 1 line MUX는 4개의 입력 중 하나를 선택하여 출력하는 회로이다. 선택 신호가 이 선택을 하도록 만드는 입력이다. 이는 데이터 선택, 제어 시스템과 같은 곳에서 응용할 수 있다.

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

**8.**

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

1 to 4 line deMUX의 simulation 결과는 아래와 같다.



1 to 4 line deMUX의 truth map은 아래와 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| input | | output | | | |
| s1 | s0 | o1 | o2 | o3 | o4 |
| 0 | 0 | a | 0 | 0 | 0 |
| 0 | 1 | 0 | a | 0 | 0 |
| 1 | 0 | 0 | 0 | a | 0 |
| 1 | 1 | 0 | 0 | 0 | a |

1 to 4 deMUX의 k-map은 아래와 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a s0s1 | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |

o1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a s0s1 | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |

o2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a s0s1 | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |

o3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a s0s1 | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |

o4

1 to 4 deMUX의 코드는 아래와 같다.

assign o1 = a&~s0&~s1;

assign o2 = a&s0&~s1;

assign o3 = a&~s0&s1;

assign o4 = a&s0&s1;

1 to 4 deMUX는 하나의 입력 신호를 4개의 출력 중 하나로 분배하는 회로이다. 이와 같은 동작 원리로 인하여 데이터 분배 혹은 메모리 주소 디코딩에서 유용하게 사용할 수 있다.

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

**9.**

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

이번 실습에서는 decoder, encdoer, MUX, deMUX 등과 같은 신호 변환 회로에 대한 프로그램을 작성하고 결과를 확인하였다. 각 소자 별 truth table과 k-map으로 논리식을 작성하고 회로를 최적화하는 방법을 사용하였다. 추가적으로 이 보고서에서 priority encdoer까지 탐구하여 여러가지 상황의 입력에 대응할 수 있게 되었다. 이 실습으로 encoding과 decoding을 하는 방법과 하는 이유 등에 대하여 알 수 있었다.

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

**10.**

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

추가적으로 연관 디코딩 (asoociative decoding)에 대하여 탐구하자면 이는 메모리 시스템에서 사용되며 입력 주소와 관련된 데이터를 빠르게 검색하기 위하여 설계되었다. 메모리에 해당하는 특정 데이터를 찾기 위하여 디코딩을 수행하며, 이는 효율적인 데이터 탐색과 분배에 도움을 준다.

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