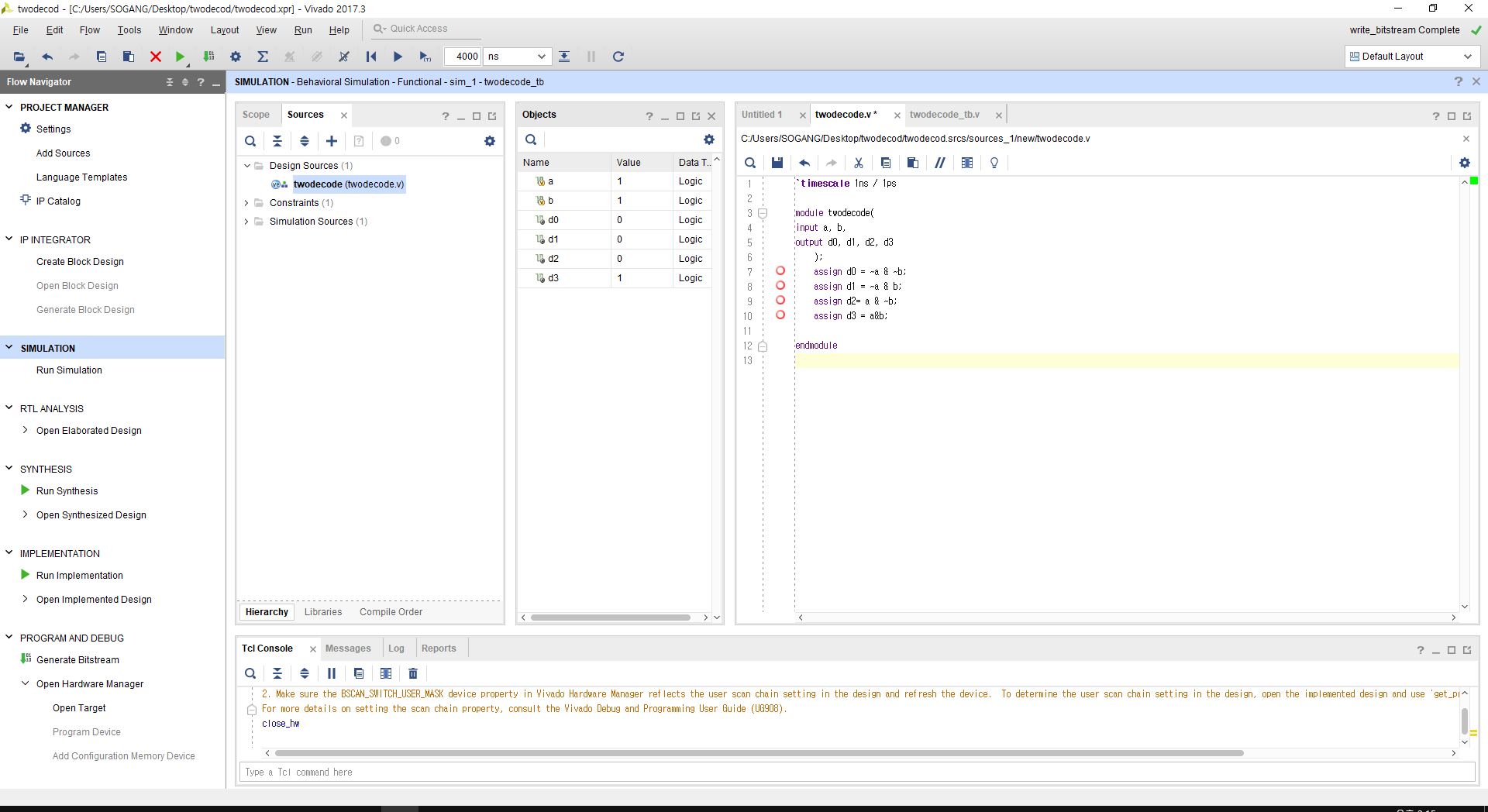
9주차 결과보고서

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

1. 2 to 4 Decoder

* Active high

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

위 그림은 2 to 4 decoder에서 N개의 bit가 입력으로 주어지고, 2^N bit만큼 출력한다.

위는 단 하나의 bit만 1인 active high 형식의 Decoder를 구현했다.

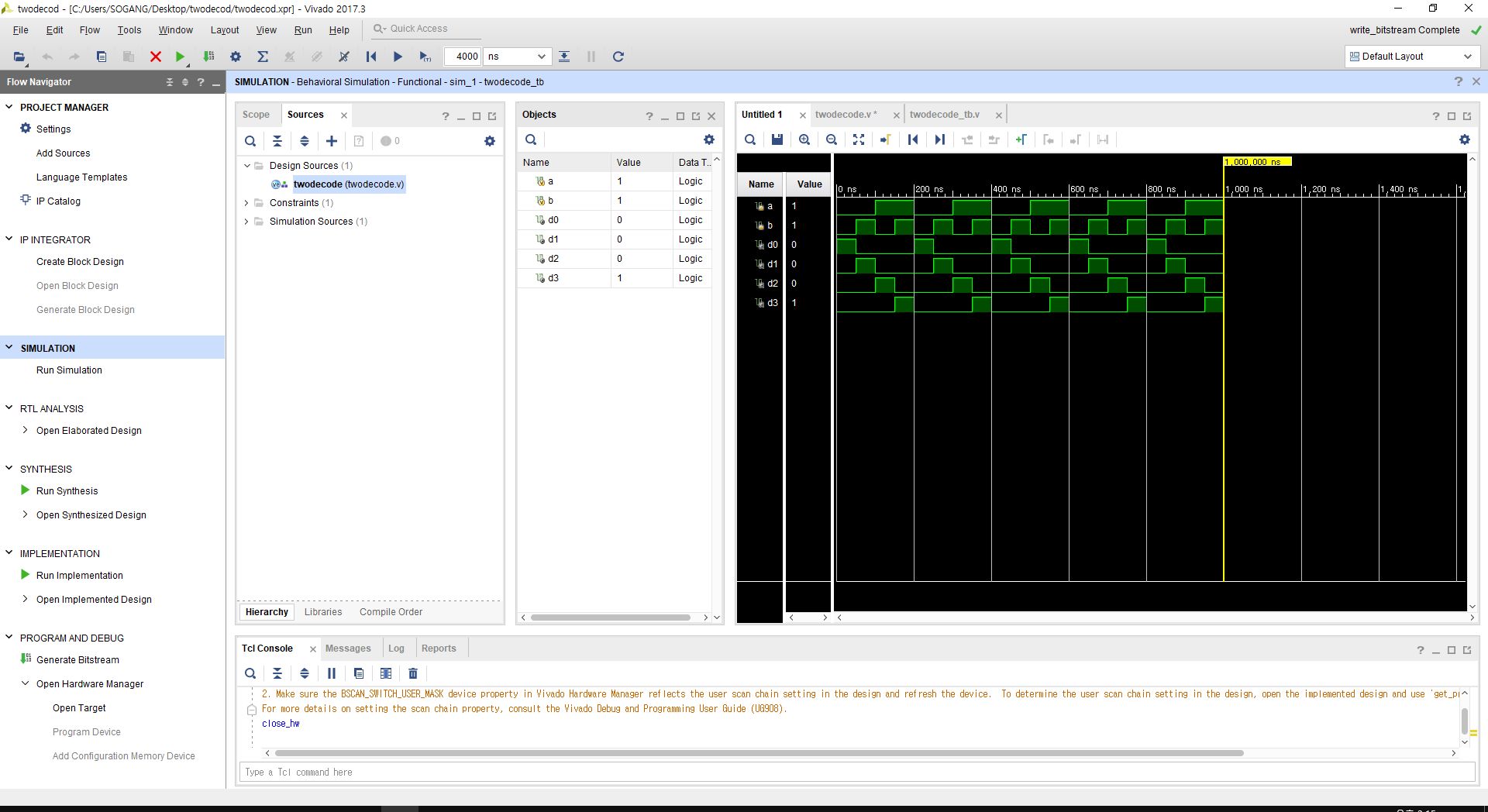
Active High의 진리표를 그리자면,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |

이 진리표를 바탕으로 카르노 맵을 작성하면

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | B  A | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | |  |  |  | | --- | --- | --- | | B  A | 0 | 1 | | 0 | 0 | 1 | | 1 | 0 | 0 | |
| D0 = A’B’ | D1 = A’B |
| |  |  |  | | --- | --- | --- | | B  A | 0 | 1 | | 0 | 0 | 0 | | 1 | 1 | 0 | | |  |  |  | | --- | --- | --- | | B  A | 0 | 1 | | 0 | 0 | 0 | | 1 | 0 | 1 | |
| D2 = AB’ | D3 = AB |

위 논리식으로 도출해 Verilog coding을 하고 난 후의 SImulation이다.



* Active Low

Active Low 같은 경우에는 위 Active High의 경우와 반대로 NAND gate를 사용해서 특정값만 0이고 나머지는 1인 경우이다.

Verilog coding은

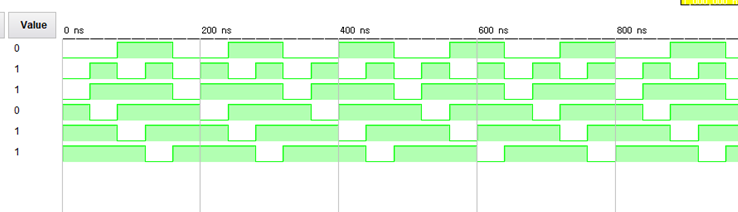
d0 = ~((~a)&(~b)); d1 = ~((~a)&b); d2 = ~(a&(~b)); d3 = ~(a&b);이다.

진리표를 작성하자면,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |

위 진리표를 이용하여, 카르노맵을 작성하면

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | B  A | 0 | 1 | | 0 | 0 | 1 | | 1 | 1 | 1 | | |  |  |  | | --- | --- | --- | | B  A | 0 | 1 | | 0 | 1 | 0 | | 1 | 1 | 1 | |
| D0 = A + B = (A’B’)’ | D1 = A+B’ = (A’B)’ |
| |  |  |  | | --- | --- | --- | | B  A | 0 | 1 | | 0 | 1 | 1 | | 1 | 0 | 1 | | |  |  |  | | --- | --- | --- | | B  A | 0 | 1 | | 0 | 1 | 1 | | 1 | 1 | 0 | |
| D2 = A’ + B = (AB’)’ | D3 = A’ + B’ = (AB)’ |



2. 4 to 2 Encoder

4 to 2 Encoder의 경우에는 디코더의 반대 개념이다. 2^n개의 입력을 받으면 n개의 output을 갖는다. 진리표를 그리자면,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | e1 | e0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |

위 진리표를 바탕으로 카르노맵을 작성해보겠다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | CD  AB | 00 | 01 | 11 | 10 | | 00 | X | 0 | X | 0 | | 01 | 1 | X | X | X | | 11 | X | X | X | X | | 10 | 1 | X | X | X | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | CD  AB | 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 = A+B | e0 = A+C |

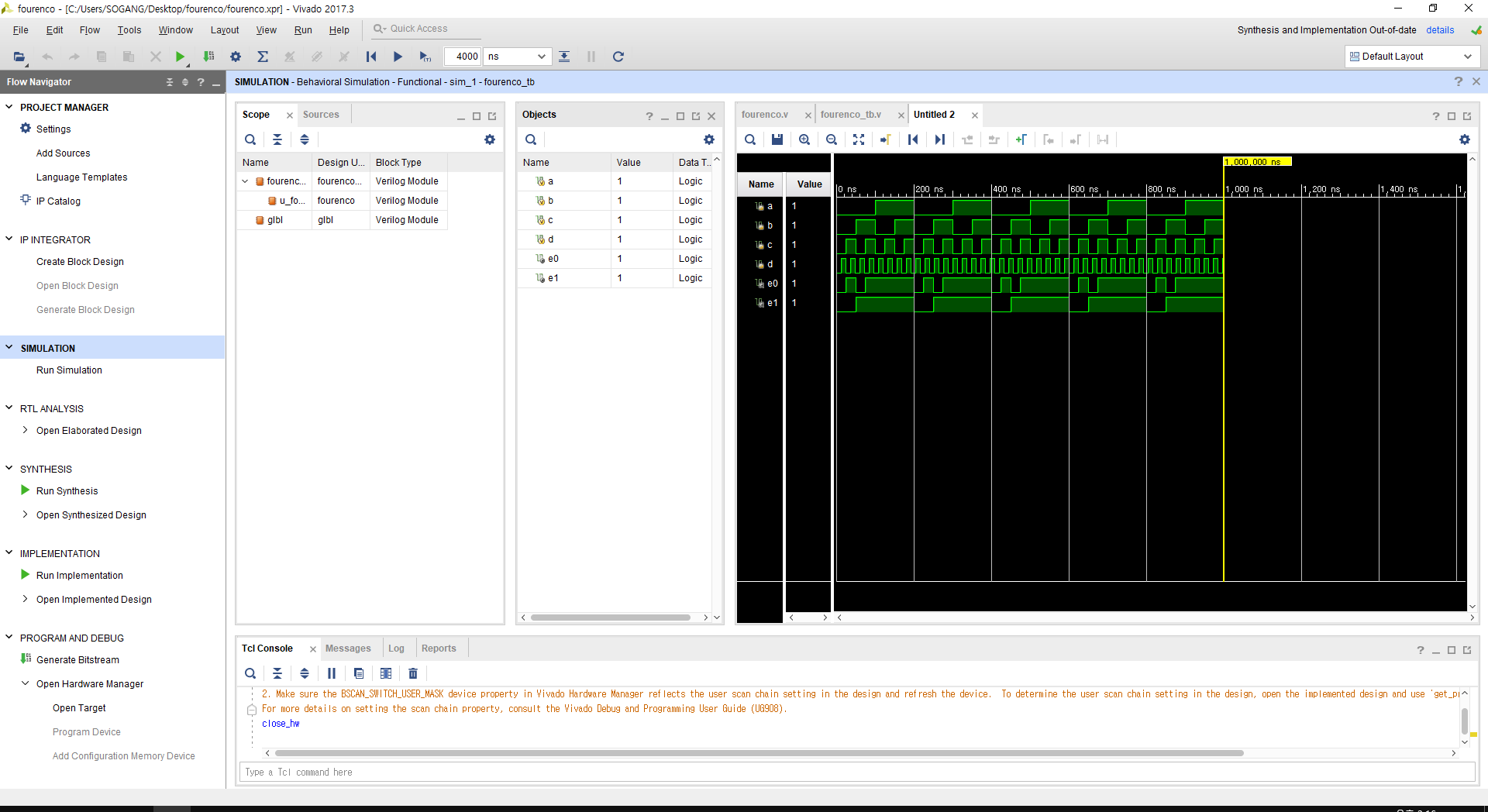
이 카르노 맵으로 Verilog 코딩을 하면,

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

* Simulation 결과



앞서, Encoder는 Decoder의 반대의 기능을 한다고 했는데, 이는 Decoder에서 출력값이었던 것을 그대로 입력값으로 받는 것을 뜻한다. 따라서, 0001이 입력되었을 때, 11이 출력되고, 0001이 입력되었을 때, 01이 출력되고, 1000이 입력되었을 때, 00이 출력된다.

3. 4 to 2 Encoder에서 입력 형태 4가지를 제외한 나머지 입력 형태는 무엇을 뜻하는지 설명하시오.

위에서 볼 수 있는 것 같이 Encoder에서 입력 형태가 단 한 개의 bit만 1이거나, 0인 경우이다. 예를 들면, 0001이나 1000을 뜻한다. 하지만 이를 제외한 경우가 나머지 입력 형태이다. 이러한 경우에는 기기에서는 오류가 발생할 가능성이 높아진다. 따라서, 이러한 오류를 방지하기 위해서 각 입력 비트들에 우선순위를 설정해야 한다. 이는 Priority Encoder이다.

4. 4 to 2 encoder의 4가지 형태가 아닌 모든 입력 형태(16가지)에 대하여 동작되는 priority encoder의 논리 회로를 구성하여라.

우선 Priority Encoder의 진리표는 이렇다. Priority 즉 우선순위는 입력 값 A부터 D까지 낮아지게 했다. V 값은 실습 때도 볼 수 있듯이, 입력 중 하나라도 1이 있으면 1이 나타난다. 만약 V 값이 0이라면, e1과 e0값은 don’t care가 된다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E1 | E0 | V |
| 0 | 0 | 0 | 0 | X | X | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

이를 이용하여 e1, e0, V 값의 카르노맵을 작성해보자.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | CD  AB | 00 | 01 | 11 | 10 | | 00 | X | 0 | 0 | 0 | | 01 | 1 | 1 | 1 | 1 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | CD  AB | 00 | 01 | 11 | 10 | | 00 | X | 0 | 1 | 1 | | 01 | 0 | 0 | 0 | 0 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 | |
| e1 = A + B | e0 = A + B’C |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | CD  AB | 00 | 01 | 11 | 10 | | 00 | 0 | 1 | 1 | 1 | | 01 | 1 | 1 | 1 | 1 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 | | |
| V = A + B + C + D | |

따라서 Verilog Code는

*module prior(*

*input a, b, c, d,*

*output e1, e0, v);*

*assign e1 = a|b;*

*assign e0 = a|(~b&c); assign v = a|b|c|d;*

*endmodule*

이다.

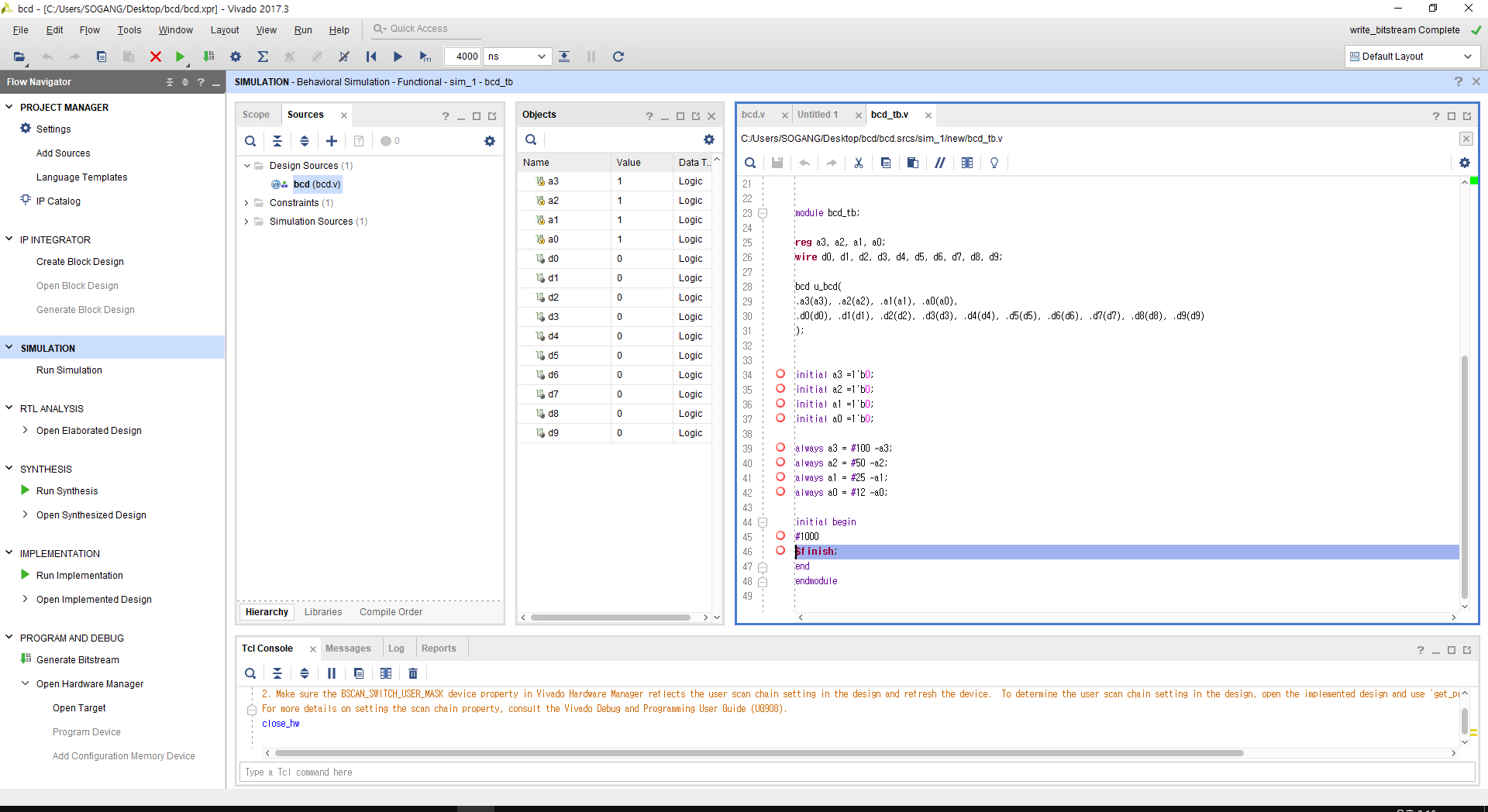
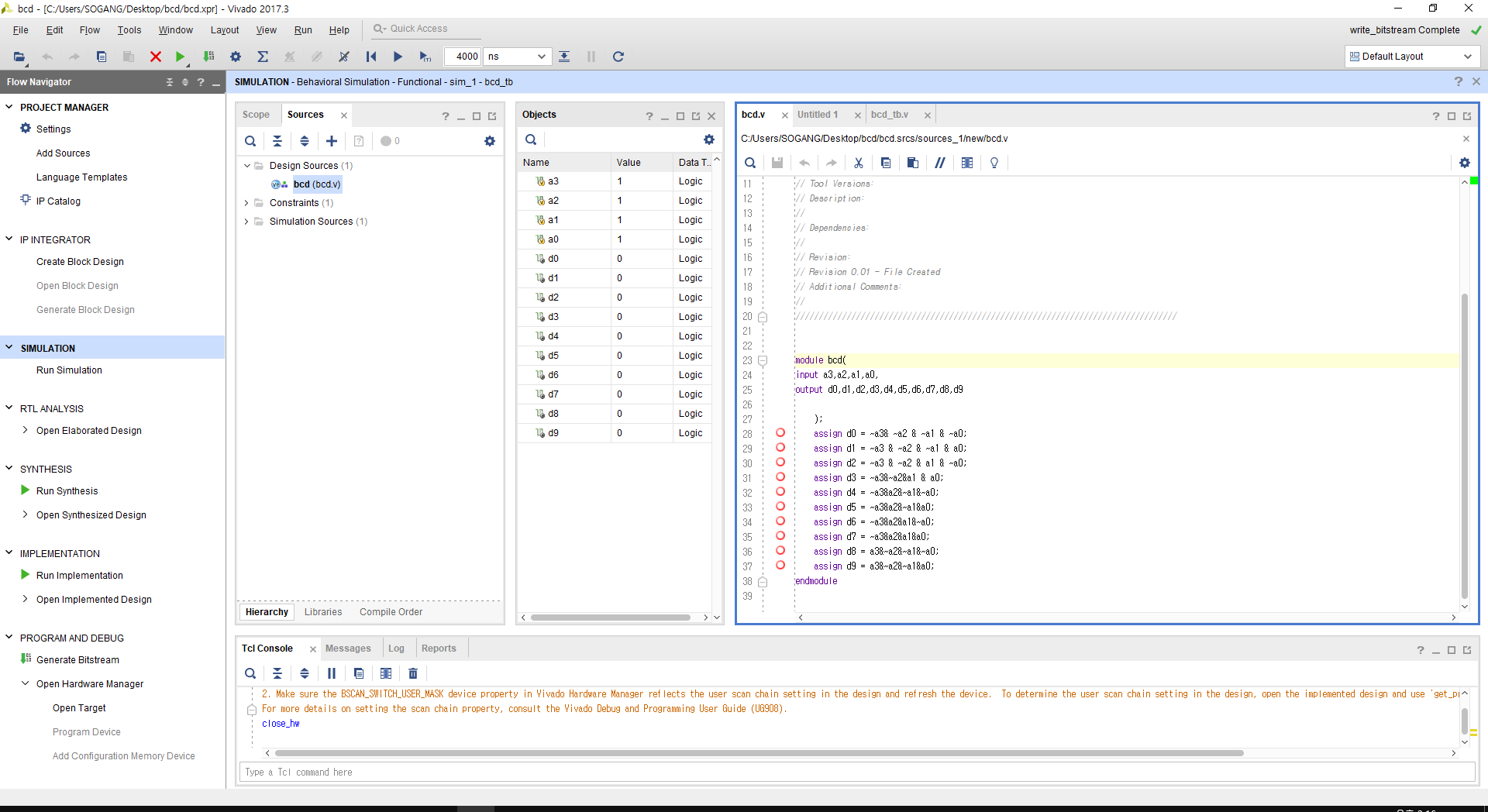
5. BCD to Decimal decoder

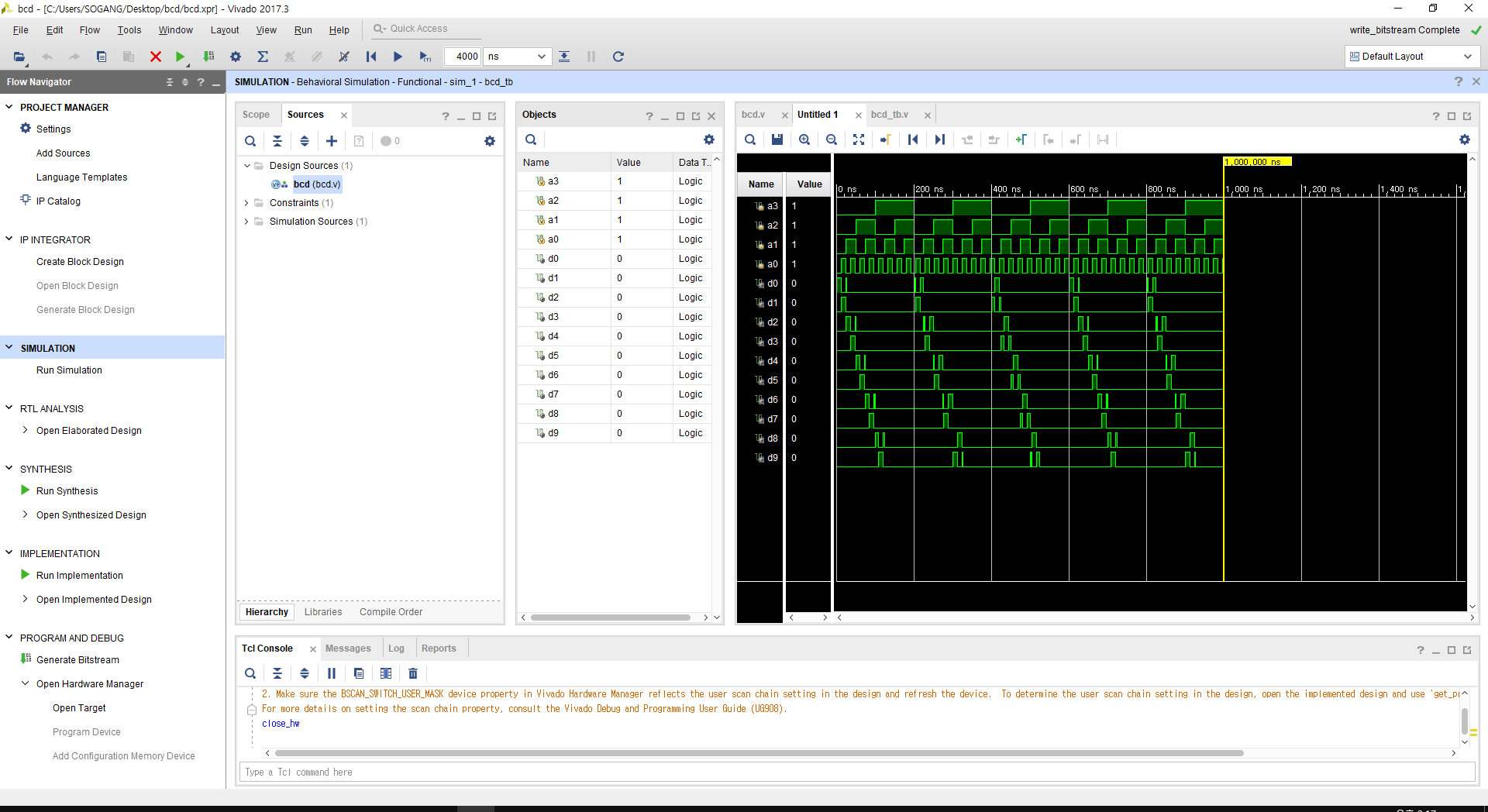
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A3 | A2 | A1 | A0 | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | X | X | X | X | X | X | X | X | X | X |
| 1 | 0 | 1 | 1 | X | X | X | X | X | X | X | X | X | X |
| 1 | 1 | 0 | 0 | X | X | X | X | X | X | X | X | X | X |
| 1 | 1 | 0 | 1 | X | X | X | X | X | X | X | X | X | X |
| 1 | 1 | 1 | 0 | X | X | X | X | X | X | X | X | X | X |
| 1 | 1 | 1 | 1 | X | X | X | X | X | X | X | X | X | X |

위를 카르노 맵을 그리자면,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 1 | 0 | 0 | 0 | | 01 | 0 | 0 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 0 | 0 | 0 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 0 | 1 | 0 | 0 | | 01 | 0 | 0 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 0 | 0 | 0 | |
| D0 = A0’A1’A2’A3’ | D1 = A0A1’A2’A3’ |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 0 | 0 | 0 | 1 | | 01 | 0 | 0 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 0 | 0 | 0 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 0 | 0 | 1 | 0 | | 01 | 0 | 0 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 0 | 0 | 0 | |
| D2 = A0’A1A2’A3’ | D3 = A0A1A2’A3’ |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 0 | 0 | 0 | 0 | | 01 | 1 | 0 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 0 | 0 | 0 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 0 | 0 | 0 | 0 | | 01 | 0 | 1 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 0 | 0 | 0 | |
| D4 = A0’A1’A2A3’ | D5 = A0A1’A2A3’ |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 0 | 0 | 0 | 0 | | 01 | 0 | 0 | 0 | 1 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 0 | 0 | 0 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 0 | 0 | 0 | 0 | | 01 | 0 | 0 | 1 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 0 | 0 | 0 | |
| D6 = A0’A1A2A3’ | D7 = A0A1A2A3’ |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 0 | 0 | 0 | 0 | | 01 | 0 | 0 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 1 | 0 | 0 | 0 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | A1A0  A3A2 | 00 | 01 | 11 | 10 | | 00 | 0 | 0 | 0 | 0 | | 01 | 0 | 0 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 1 | 0 | 0 | |
| D8 = A0’A1’A2’A3 | D9 = A0A1’A2’A3 |

이를 바탕으로 코딩한 Verilog code와 Simulation은 이렇다.





6. Encoder와 decoder의 주요 응용

Encoder의 경우, 데이터를 압축하기 위한 용도로 자주 사용된다 우리가 파일을 압축할 때, 동영상을 압축할 때, 그리고 통신 기술에서 암호화하며, 부호화하는 과정에서 사용이 가능하다. 반대로 Decoder의 경우, 이러한 압축된 파일과 동영상을 원본 파일로 복원하는 과정에서 사용되기도 하고, 통신 기술에서 암호화되어있는, 부호화되어있는 통신 신호들을 복원하는 역할을 할 수 있다. Encoder와 Decoder는 반대의 역할을 하게 된다. 따라서 송신 측에는 Encoder가 수신 측에는 Decoder가 쓰이는 경우가 많다.

7. 4 to 1 line MUX

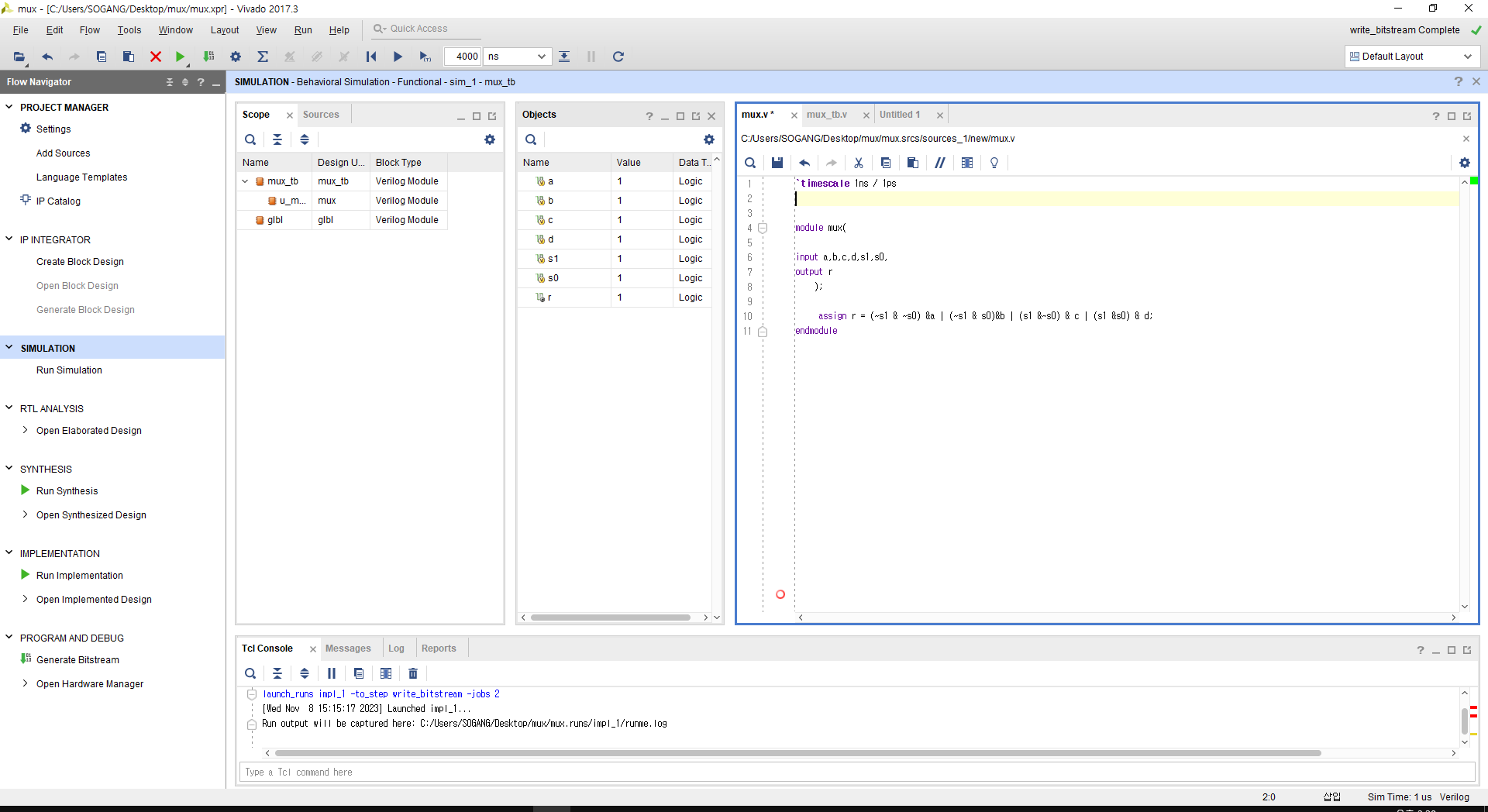
멀티플렉서는 여러 개의 입력 신호가 들어왔을 때 입력 신호를 한 개만 선택하는 논리 회로인데,

s1과 s0의 값을 통해서 몇변 째 신호를 입력 신호로 취할 지 선택한다.

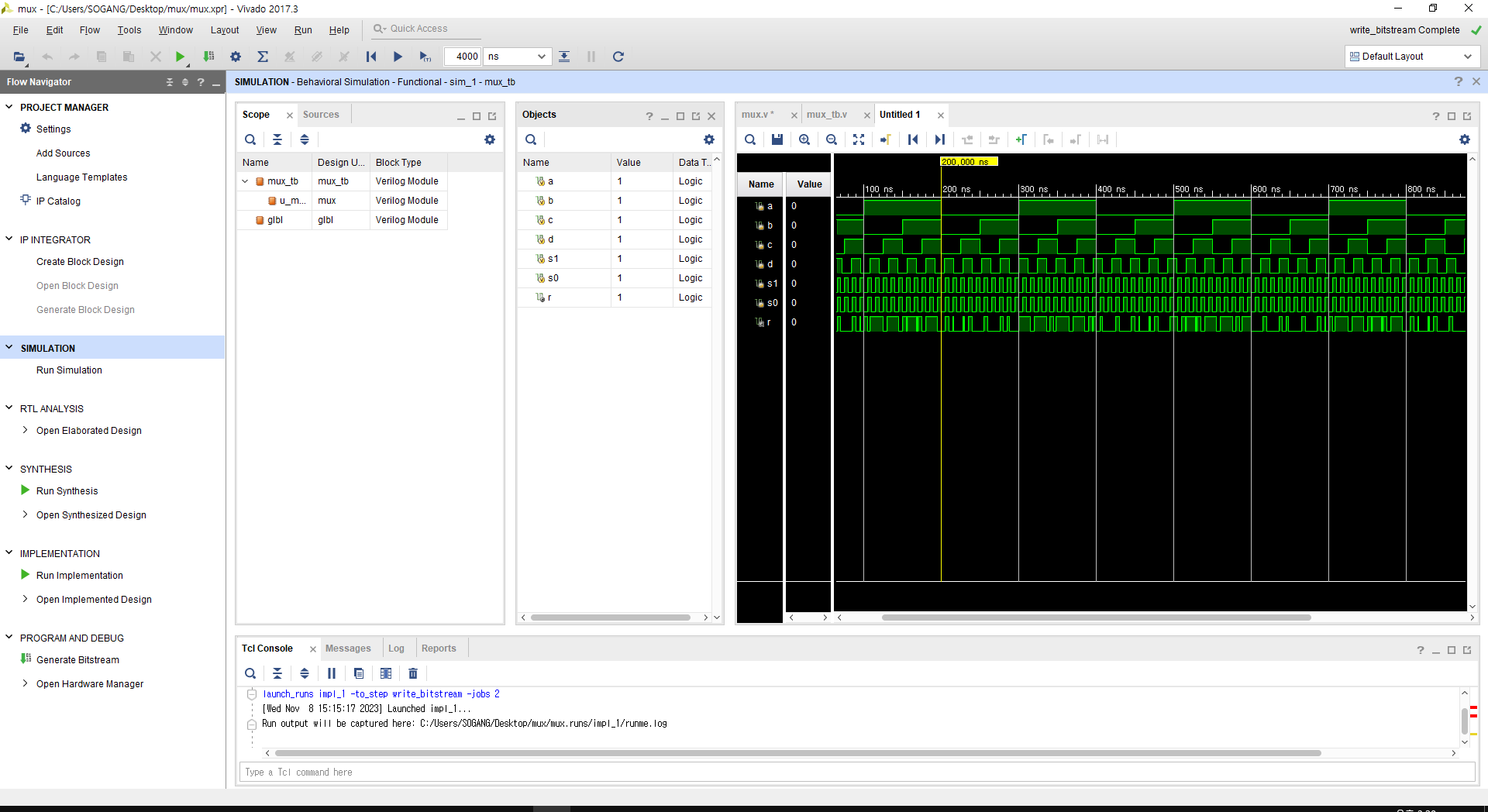
s1과 s0가 00이면 A, s1과 s0이 01이라면 B를 선택한 것을 알 수 있다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **s1** | **s0** | **r** |
| 0 | X | X | X | 0 | 0 | 0 |
| 1 | X | X | X | 0 | 0 | 1 |
| X | 0 | X | X | 0 | 1 | 0 |
| X | 1 | X | X | 0 | 1 | 1 |
| X | X | 0 | X | 1 | 0 | 0 |
| X | X | 1 | X | 1 | 0 | 1 |
| X | X | X | 0 | 1 | 1 | 0 |
| X | X | X | 1 | 1 | 1 | 1 |

위 설명을 기반으로 한 Verilog code 및 시뮬레이션이다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

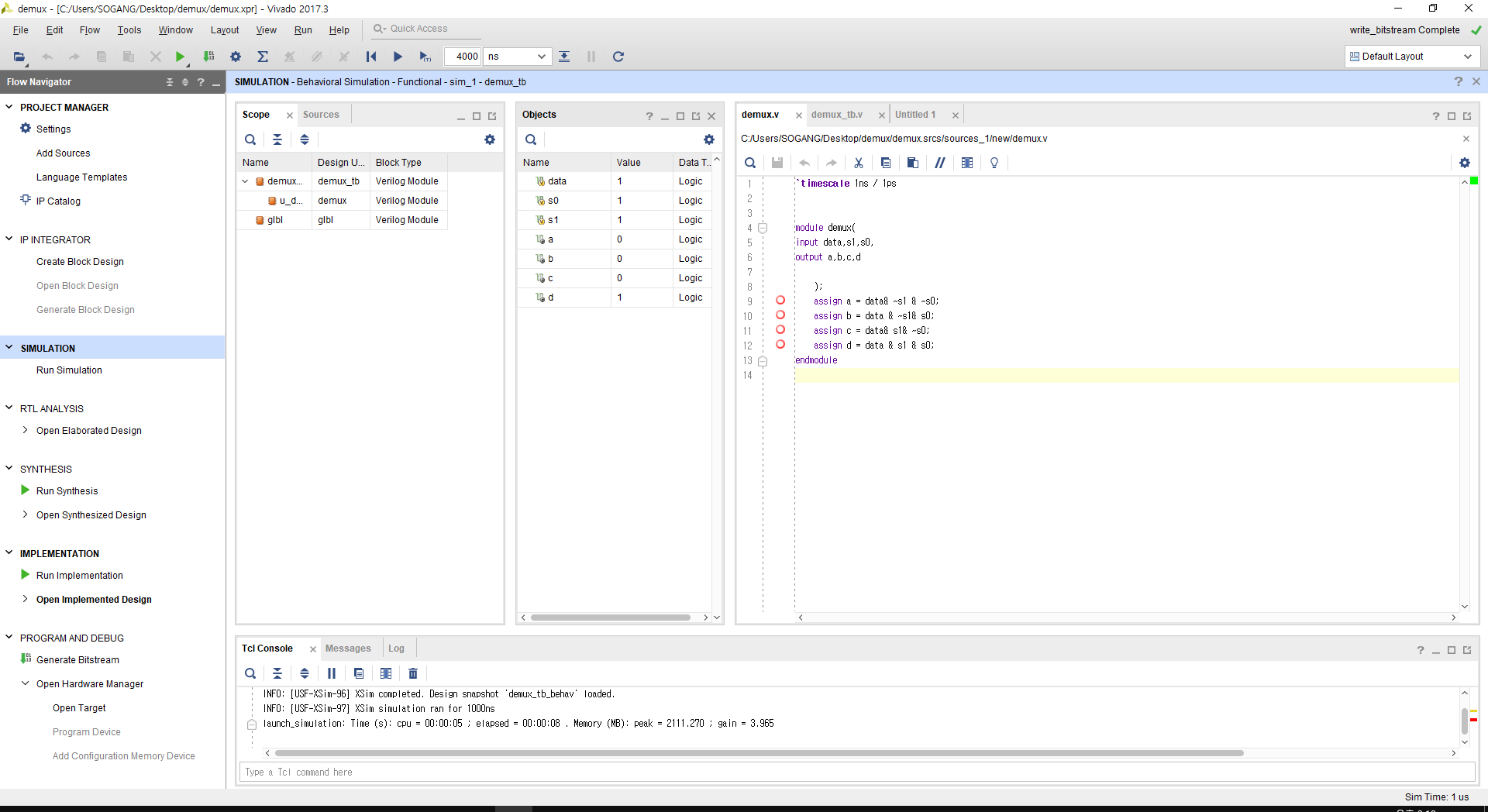
자동 생성된 설명



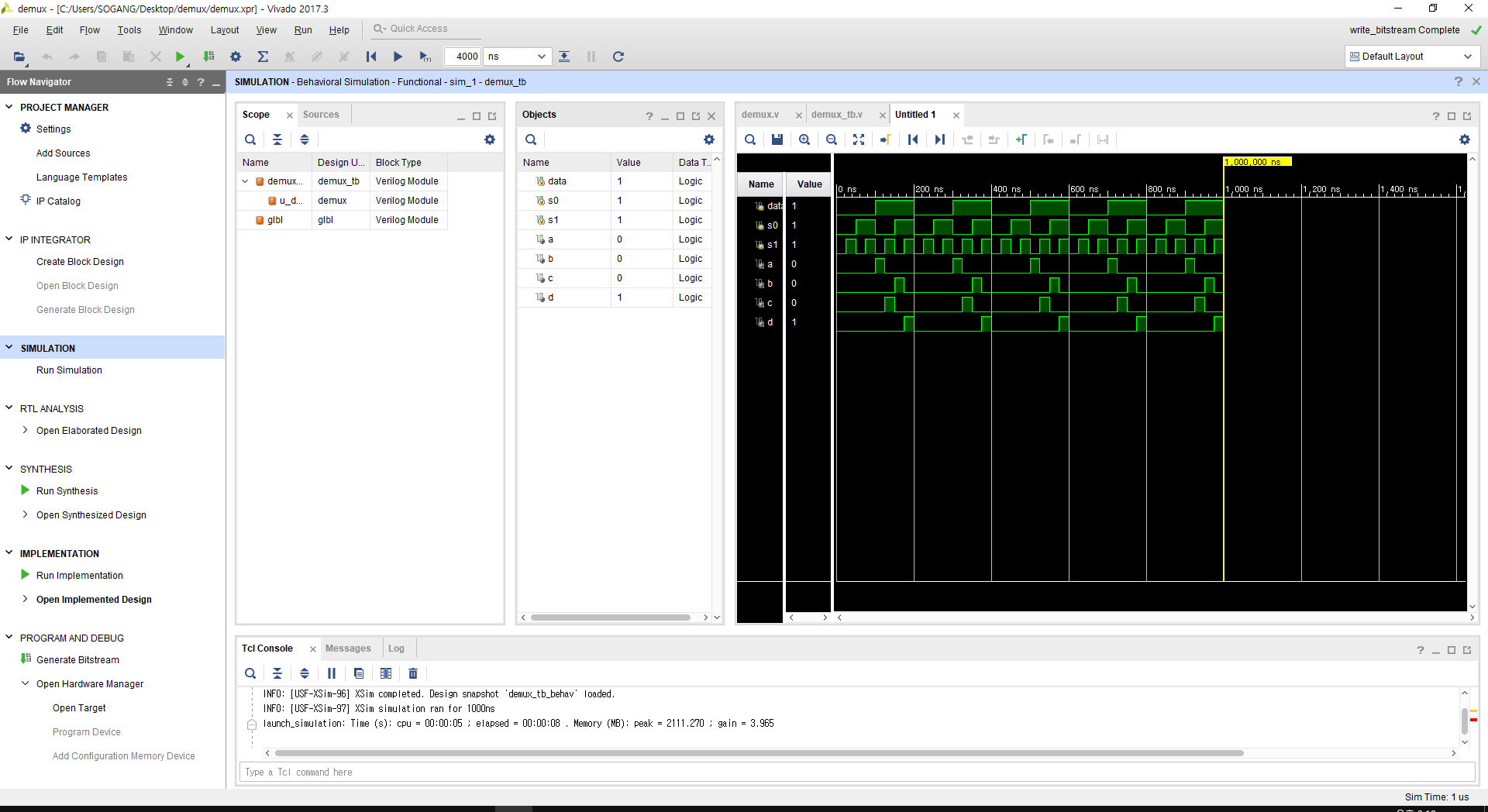
8. 1-to-4-line deMUX

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| s1 | s0 | data | a | b | c | d |
| 0 | 0 | 0 | 0 | X | X | X |
| 0 | 1 | 0 | X | 0 | X | x |
| 1 | 0 | 0 | X | X | 0 | X |
| 1 | 1 | 0 | X | X | x | 0 |
| 0 | 0 | 1 | 1 | X | X | X |
| 0 | 1 | 1 | X | 1 | X | X |
| 1 | 0 | 1 | X | X | 1 | X |
| 1 | 1 | 1 | x | X | 0 | 1 |

deMux는 반대로 s1과 s0 값에 따라, data가 주어진다면, 만약 Mux에 입력된 신호라면, 이를 복원하는 역할이다. 예를 들어, s1과 s0가 각각 00이라면, 앞에 설명한 것과 마찬가지로 a를 선택을 할 것이다. 그 후, data가 0이기 때문에 출력값의 a는 0이 된다. 나머지 신호는 알 수 없으므로 don’t care로 출력한다. 아래는 위 진리표와 기능을 바탕으로 한 Verilog 코딩이다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명



9. 결과 검토 및 논의 사항.

이번 주차 실습 때 Decoder와 Encoder 및, MUX와 DeMUX를 구현하였다. Decoder의 경우 n개의 입력을 받고 2^n개를 출력하는 형태로 구현했는데, 이를 active high와 low를 나눠서 구현한 점이 인상깊었다. active high와 low의 차이는 출력 신호가 1이냐 0이냐 하는 차이점과 동일하다. 따라서 AND 연산으로 구현한 것과 NAND를 취해줘 구현한 결과를 비교해서 학습할 수 있었다. Active High와 Active Low를 나눠서 하는 이유와 각각의 장점과 단점, 활용 사례를 공부해볼 필요가 있다고 느꼈다.

10. 추가 이론 조사 및 작성.

프로그래밍 개념 중 인코딩과 디코딩이라는 개념이 있다. 이는 컴퓨터가 이해할 수 있는 이진법으로 변환하는 것 또는 이진법 코드를 문자로 변환하는 것이다. 이 컴퓨터가 이해할 수 있는 이진법 코드(binary code)에는 대표적으로 ASCII Code, 유니코드 등이 있다. 이중 Unicode는 전세계의 모든 문자를 컴퓨터에서 표현하기 위한 코드이다. 대표적인 Unicode 인코딩 방식으로 UTF-8이 있다. 이처럼, 컴퓨터가 이해하기 쉬운 코드로 변환할 때 encoding, 즉 문자를 code화 하는 것을 encoder가, 컴퓨터의 code를 user가 보기 쉬운 문자로 표현하게 하는 것을 decoder가 맡아서 진행한다.

스크린샷, 텍스트, 다채로움, 디자인이(가) 표시된 사진

자동 생성된 설명