1. 9주차 결과보고서

|  |
| --- |
| - 2X4 Decoder  - 4X2 Encoder  - BCD to Decimal Decoder  - Encoder & Decoder 응용  - 8X1 MUX  - 1X4 DEMUX->4X1 DECODER  - 결과 검토 및 논의 사항  - 추가 이론 |

**20141196 김성희**

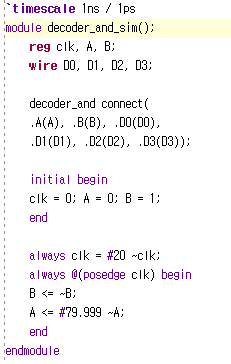
1. 2X4 Decoder
   1. **가. AND gate**

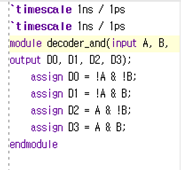
**- Truth table : 이론**

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

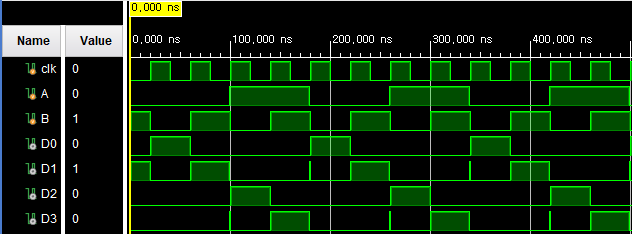
**- Karnaugh map**

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

**- code**

design source code simulation code

**-simulation**



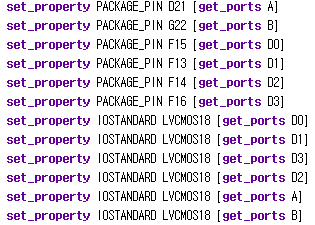
**< truth table> - 결과**

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

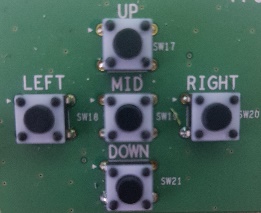
**- FPGA**

업로드:

Design Sources경로에 code 짜기 -> Run Synthesis & Run Implementation -> Constraints경로에 .xdc 파일 추가하기 -> Open Implemented Design 클릭 후 Constraints Wizard 클릭(Define Target, .xdc파일 타겟 설정) -> Window 탭에서 I/O ports 클릭 -> 원하는 pin 선택 및 LVCMOS18(I/O Std) 선택 -> 저장 후 .xdc reload(아래 그림처럼 코드가 자동으로 짜서 나온다.) -> Generate Bitstream 클릭(Synthesis, Implementation도 자동으로 실행) 후에 Open Hardware Manager 클릭 -> Open Target 클릭 후 Auto Connect 클릭 -> Program Device 클릭 -> FPGA에 업로드 끝

<- pin 배정

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **사진**  **(switch 눌렀을 때의 결과)** |  |  |  |  |
| **AB** | **00** | **01** | **10** | **11** |
| **PUSH**  **SWITCH** | **NONE** | **RIGHT** | **LEFT** | **RIGHT**  **LEFT** |

 LEFT = input A, RIGHT = input B

위 표에서 PUSH SWITCH의 switch를 누르면 해당되는 사진처럼 LED가 켜진다.

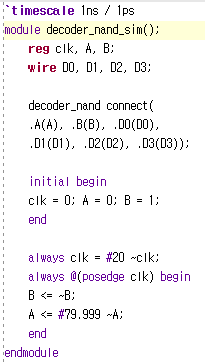
**나. NAND gate**

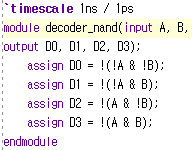
**- Truth table : 이론**

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

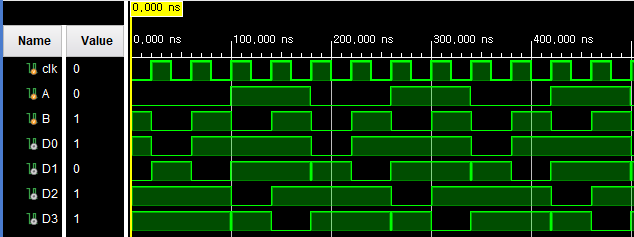
**- Karnaugh map**

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

**- code**

design source code simulation code

**-simulation**



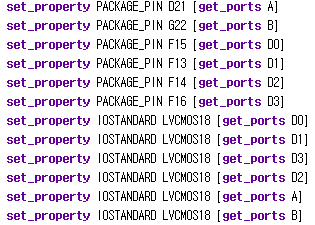
**< truth table> - 결과**

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

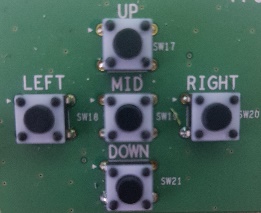
**- FPGA**

업로드:

Design Sources경로에 code 짜기 -> Run Synthesis & Run Implementation -> Constraints경로에 .xdc 파일 추가하기 -> Open Implemented Design 클릭 후 Constraints Wizard 클릭(Define Target, .xdc파일 타겟 설정) -> Window 탭에서 I/O ports 클릭 -> 원하는 pin 선택 및 LVCMOS18(I/O Std) 선택 -> 저장 후 .xdc reload(아래 그림처럼 코드가 자동으로 짜서 나온다.) -> Generate Bitstream 클릭(Synthesis, Implementation도 자동으로 실행) 후에 Open Hardware Manager 클릭 -> Open Target 클릭 후 Auto Connect 클릭 -> Program Device 클릭 -> FPGA에 업로드 끝

<-pin 배정

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **사진**  **(switch 눌렀을 때의 결과)** |  |  |  |  |
| **AB** | **00** | **01** | **10** | **11** |
| **PUSH**  **SWITCH** | **NONE** | **RIGHT** | **LEFT** | **RIGHT**  **LEFT** |

 LEFT = input A, RIGHT = input B

위 표에서 PUSH SWITCH의 switch를 누르면 해당되는 사진처럼 LED가 켜진다.

(LD1: D0, LD2: D1, LD3: D2, LD4: D3)

1. 2X4 Encoder
   1. **가. Encoder**

**- Truth table : 이론**

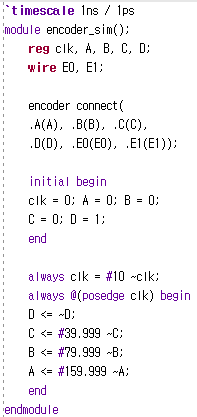
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | |
| **A** | **B** | **C** | **D** | **E0** | **E1** |
| **0** | **0** | **0** | **1** | **0** | **0** |
| **0** | **0** | **1** | **0** | **0** | **1** |
| **0** | **1** | **0** | **0** | **1** | **0** |
| **1** | **0** | **0** | **0** | **1** | **1** |

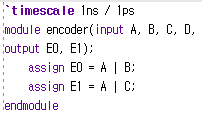
**- Karnaugh map**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 0 | 1 | 0 | 1 | | 01 | 0 | 0 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 0 | 0 | 0 | 0 |   **E0=A’BC’D’+AB’C’D’** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 0 | 0 | 0 | 1 | | 01 | 0 | 0 | 0 | 0 | | 11 | 0 | 0 | 0 | 0 | | 10 | 1 | 0 | 0 | 0 |   **E1=A’B’CD’+AB’C’D’** |

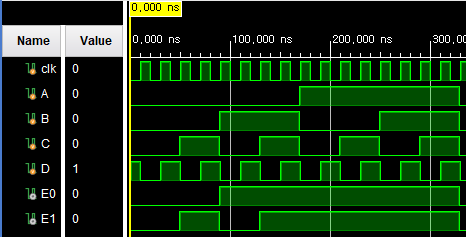
**카르노 맵을 통해서 나온 논리식은 위와 같지만 본 실험에서는 OR gate를 이용하여,**

**E0=A+B, E1=A+C와 같은 논리식으로 구현하기로 한다. (input이 정확히 온다는 가정하에)**

**- code**

design source code simulation code

**-simulation**



**< truth table> - 결과**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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** | **0** | **1** |
| **0** | **1** | **0** | **0** | **1** | **0** |
| **0** | **1** | **0** | **1** | **1** | **0** |
| **0** | **1** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** |
| **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** |

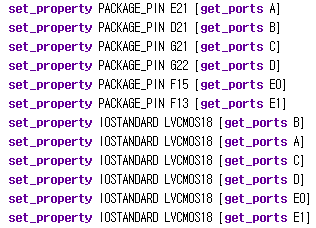
**4가지 input 형태 이외에 input들에 대해서 결과값이 어떻게 나타나는지 살펴보자.**

0000은 0001과 같다. 0011은 0010과 같다. 0101은 0100과 같다. 0110~0111, 1001~1111은 1000과 같다. 즉 0000~0001은 00값을, 0010~0011은 01값을, 0100~0101은 10값을, 0110~1111은 11값을 내놓는다. Input을 십진수로 표시했을 때0부터 15까지 순서대로 00, 01, 10, 11값을 가진다. 이 다음 실험인 priority encoder와 유사한 output을 가진다.

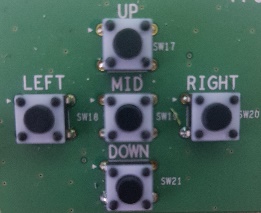
**- FPGA**

업로드:

Design Sources경로에 code 짜기 -> Run Synthesis & Run Implementation -> Constraints경로에 .xdc 파일 추가하기 -> Open Implemented Design 클릭 후 Constraints Wizard 클릭(Define Target, .xdc파일 타겟 설정) -> Window 탭에서 I/O ports 클릭 -> 원하는 pin 선택 및 LVCMOS18(I/O Std) 선택 -> 저장 후 .xdc reload(아래 그림처럼 코드가 자동으로 짜서 나온다.) -> Generate Bitstream 클릭(Synthesis, Implementation도 자동으로 실행) 후에 Open Hardware Manager 클릭 -> Open Target 클릭 후 Auto Connect 클릭 -> Program Device 클릭 -> FPGA에 업로드 끝

<- pin 배정

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **사진**  **(switch 눌렀을 때의 결과)** |  |  |  |  |
| **ABCD** | **0001** | **0010** | **0100** | **1000** |
| **PUSH**  **SWITCH** | **RIGHT** | **MID** | **LEFT** | **UP** |

 UP = input A, LEFT = input B, MID = input C, RIGHT = input D

위 표에서 PUSH SWITCH의 switch를 누르면 해당되는 사진처럼 LED가 켜진다.

(LD1: E0, LD2: E1)

Simulation과 truth table에서 4개의 input이외의 input들에 대한 output에 대해서 이야기했던 대로 fpga에서도 NONE->0001, (MID, RIGHT)->0010, (LEFT, RIGHT)->0100, { 나머지 }->1000과 같은 결과로 LED가 켜진다.

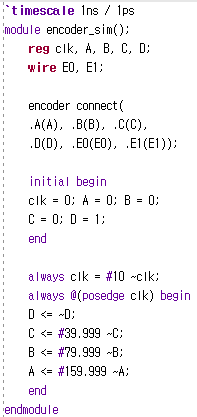
* 1. **나. Priority Encoder**

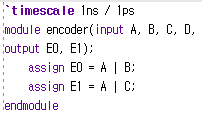
**- Truth table : 이론**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | |
| **A** | **B** | **C** | **D** | **E0** | **E1** |
| **0** | **0** | **0** | **1** | **0** | **0** |
| **0** | **0** | **1** | **X** | **0** | **1** |
| **0** | **1** | **X** | **X** | **1** | **0** |
| **1** | **X** | **X** | **X** | **1** | **1** |
| **0** | **0** | **0** | **0** | **X** | **X** |

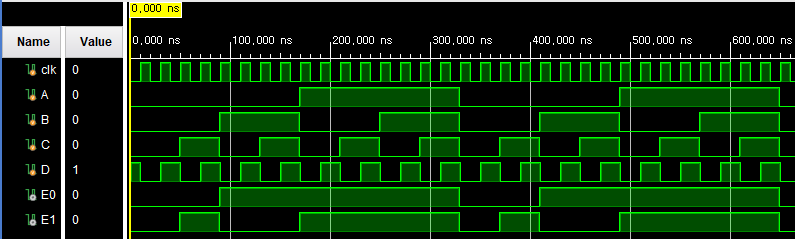
**- Karnaugh map**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | X | 1 | 1 | 1 | | 01 | 0 | 1 | 1 | 1 | | 11 | 0 | 1 | 1 | 1 | | 10 | 0 | 1 | 1 | 1 |   **E0=A+B** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | X | 0 | 1 | 1 | | 01 | 0 | 0 | 1 | 1 | | 11 | 1 | 0 | 1 | 1 | | 10 | 1 | 0 | 1 | 1 |   **E1=A+B’C** |

**- code**

design source code simulation code

**-simulation**



**< truth table> - 결과**

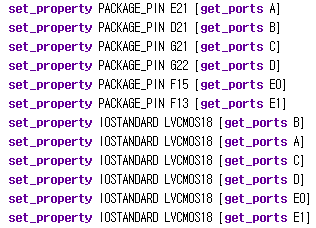
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | |
| **A** | **B** | **C** | **D** | **E0** | **E1** |
| **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **1** | **0** | **0** |
| **0** | **0** | **1** | **0** | **0** | **1** |
| **0** | **0** | **1** | **1** | **0** | **1** |
| **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** |

**이전 실험인 encoder에서의 truth table과의 차이점은 0110, 0111의 결과값이 11에서 10으로 바뀌었다는 것이다.**

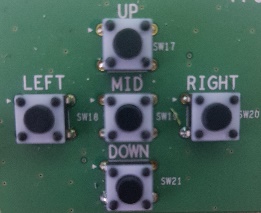
**- FPGA**

업로드:

Design Sources경로에 code 짜기 -> Run Synthesis & Run Implementation -> Constraints경로에 .xdc 파일 추가하기 -> Open Implemented Design 클릭 후 Constraints Wizard 클릭(Define Target, .xdc파일 타겟 설정) -> Window 탭에서 I/O ports 클릭 -> 원하는 pin 선택 및 LVCMOS18(I/O Std) 선택 -> 저장 후 .xdc reload(아래 그림처럼 코드가 자동으로 짜서 나온다.) -> Generate Bitstream 클릭(Synthesis, Implementation도 자동으로 실행) 후에 Open Hardware Manager 클릭 -> Open Target 클릭 후 Auto Connect 클릭 -> Program Device 클릭 -> FPGA에 업로드 끝

<- pin 배정

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **사진**  **(switch 눌렀을 때의 결과)** | |  |  |  |  |
| **SWITCH**  **ON**  **OFF** | **UP(A)** | **OFF** | **OFF** | **OFF** | **ON** |
| **LEFT(B)** | **OFF** | **OFF** | **ON** | **don’t care** |
| **MID(C)** | **OFF** | **ON** | **don’t care** | **don’t care** |
| **RIGHT(D)** | **don’t care** | **don’t care** | **don’t care** | **don’t care** |

 UP = input A, LEFT = input B, MID = input C, RIGHT = input D

위 표에서 PUSH SWITCH의 switch를 누르면 해당되는 사진처럼 LED가 켜진다.

(LD1: E0, LD2: E1)

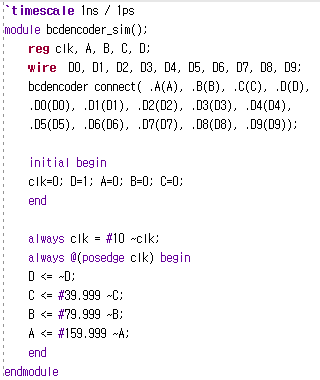
1. BCD to Decimal Decoder

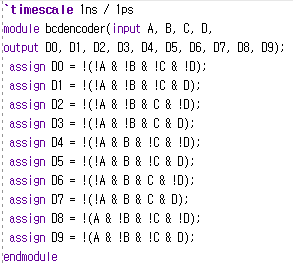
**- Truth table : 이론**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | | | | | | | | | |
| **A** | **B** | **C** | **D** | **D0** | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** | **D8** | **D9** |
| **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |

**- Karnaugh map**

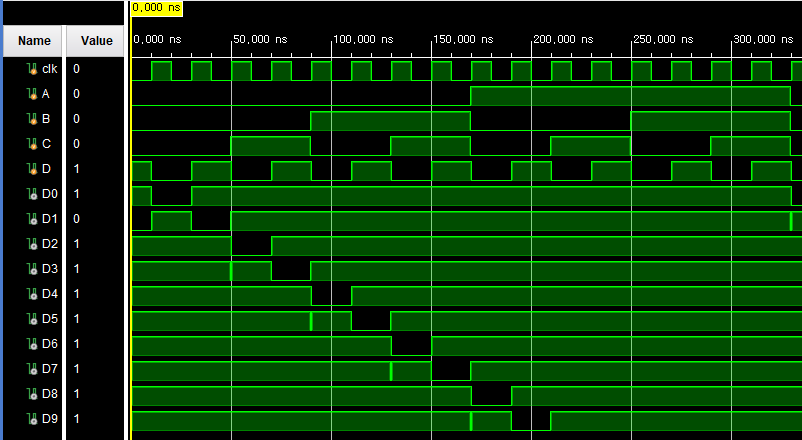
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 0 | 1 | 1 | 1 | | 01 | 1 | 1 | 1 | 1 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 |   **D0=A+B+C+D=(A’B’C’D’)’** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 1 | 1 | 1 | 1 | | 01 | 0 | 1 | 1 | 1 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 |   **D1=A+B+C+D’=(A’B’C’D)’** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 1 | 1 | 1 | 1 | | 01 | 1 | 1 | 1 | 1 | | 11 | 1 | 1 | 1 | 1 | | 10 | 0 | 1 | 1 | 1 |   **D2=A+B+C’+D=(A’B’CD’)’** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 1 | 1 | 1 | 1 | | 01 | 1 | 1 | 1 | 1 | | 11 | 0 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 |   **D3=A+B+C’+D’=(A’B’CD)’** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 1 | 0 | 1 | 1 | | 01 | 1 | 1 | 1 | 1 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 |   **D4=A+B’+C+D=(A’BC’D’)’** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 1 | 1 | 1 | 1 | | 01 | 1 | 0 | 1 | 1 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 |   **D5=A+B’+C+D’=(A’BC’D)’** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 1 | 1 | 1 | 1 | | 01 | 1 | 1 | 1 | 1 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 0 | 1 | 1 |   **D6=A+B’+C’+D=(A’BCD’)’** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 1 | 1 | 1 | 1 | | 01 | 1 | 1 | 1 | 1 | | 11 | 1 | 0 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 |   **D7=A+B’+C’+D’=(A’BCD)’** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 1 | 1 | 1 | 0 | | 01 | 1 | 1 | 1 | 1 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 |   **D8=A’+B+C+D=(AB’C’D’)’** | |  |  |  |  |  | | --- | --- | --- | --- | --- | | AB  CD | 00 | 01 | 11 | 10 | | 00 | 1 | 1 | 1 | 1 | | 01 | 1 | 1 | 1 | 0 | | 11 | 1 | 1 | 1 | 1 | | 10 | 1 | 1 | 1 | 1 |   **D9=A’+B+C+D’=(AB’C’D)’** |

**- code**



design source code simulation code

**-simulation**



**< truth table> - 결과**

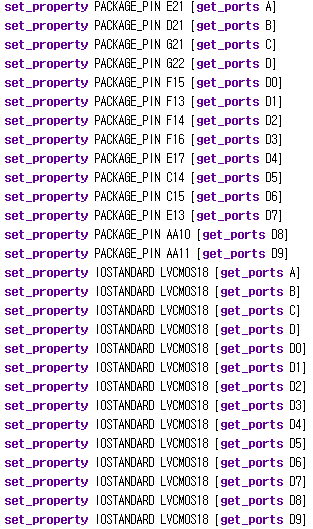
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | | | | | | | | | |
| **A** | **B** | **C** | **D** | **D0** | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** | **D8** | **D9** |
| **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |

\*0000~1001의 경우에만 의미 있는 결과 값(0~9)이 할당 되고 나머지의 경우는 0~9 어디에도 속하지 않는다.

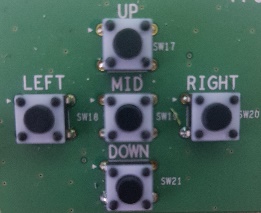
**- FPGA**

업로드:

Design Sources경로에 code 짜기 -> Run Synthesis & Run Implementation -> Constraints경로에 .xdc 파일 추가하기 -> Open Implemented Design 클릭 후 Constraints Wizard 클릭(Define Target, .xdc파일 타겟 설정) -> Window 탭에서 I/O ports 클릭 -> 원하는 pin 선택 및 LVCMOS18(I/O Std) 선택 -> 저장 후 .xdc reload(아래 그림처럼 코드가 자동으로 짜서 나온다.) -> Generate Bitstream 클릭(Synthesis, Implementation도 자동으로 실행) 후에 Open Hardware Manager 클릭 -> Open Target 클릭 후 Auto Connect 클릭 -> Program Device 클릭 -> FPGA에 업로드 끝

<- pin 배정

|  |  |  |
| --- | --- | --- |
| **사진** |  |  |
| **ABCD** | **0000** | **0001** |
| **PUSH**  **SWITCH** | **NONE** | **RIGHT** |
| **사진** |  |  |
| **ABCD** | **0010** | **0011** |
| **PUSH**  **SWITCH** | **MID** | **MID**  **RIGHT** |
| **사진** |  |  |
| **ABCD** | **0100** | **0101** |
| **PUSH**  **SWITCH** | **LEFT** | **LEFT**  **RIGHT** |
| **사진** |  |  |
| **ABCD** | **0110** | **0111** |
| **PUSH**  **SWITCH** | **LEFT**  **MID** | **LEFT**  **MID RIGHT** |
| **사진** |  |  |
| **ABCD** | **1000** | **1001** |
| **PUSH**  **SWITCH** | **UP** | **UP**  **RIGHT** |

 UP = input A, LEFT = input B, MID = input C, RIGHT = input D

위 표에서 PUSH SWITCH의 switch를 누르면 해당되는 사진처럼 LED가 켜진다.

1. Encoder & Decoder 응용

- Encoder는 주로 사람이 이해하기 쉬운 패턴을 축약해서 컴퓨터가 이해하기 쉬운 패턴으로 바꿔준다.

- Decoder는 반대로 컴퓨터가 이해하기 쉬운 패턴에서 사람이 이해하기 쉬운 패턴으로 풀어준다.

- 예를 들어 십진수 부호기와 복호기가 있다. (BCD to Decimal Decoder 실험 부분 참조)

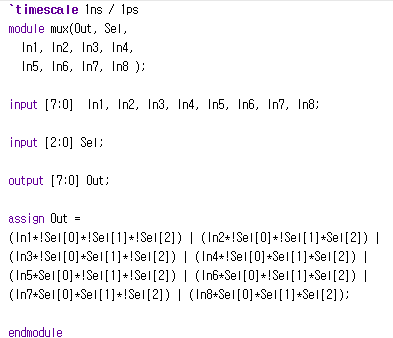
Encoder의 경우 사람이 이해하기 쉬운 0 ~ 9까지의 십진수를 0111111111, 1011111111 ~ 1111111110와 같은 input으로 넣어서 0000, 0001 ~ 1001처럼 컴퓨터가 이해하기 쉬운 패턴으로 축약해준다. (from Decimal to BCD)

Decoder의 경우 Encoder를 반대로 진행한 경우로 BCD to Decimal Decoder처럼 컴퓨터가 이해하기 쉬운 패턴에서 사람이 이해하기 쉬운 패턴인 십진수로 풀어준다. (from BCD to Decimal)

1. 8X1 MUX

**- code**

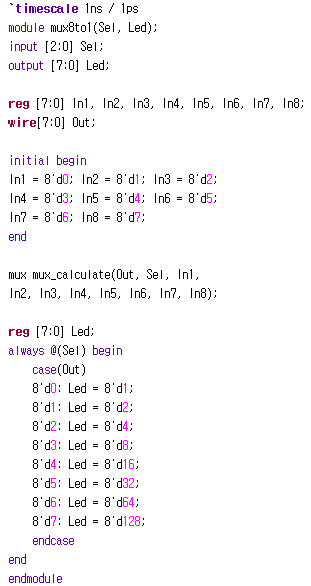
**1. mux모듈**

****

input으로 3bit 크기 Sel과 8bit 크기 In1~8을 설정  
- Sel은 3개의 selection비트로 이루어져 있다.  
- In1~8은 8비트짜리 8개의 input값이다.

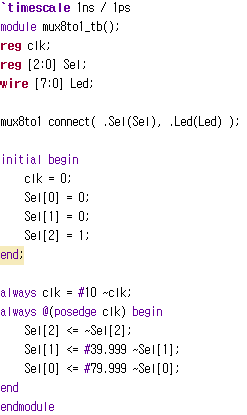
output으로 8bit 크기 Out을 설정  
- Sel[0] Sel[1] Sel[2] 패턴에 따라 Out = InN (N=1~8)이다. 즉 In1~8중 한 개를 선택해서 Out으로 넘김.

**2. mux8to1모듈 (mux모듈을 fpga와 연결해 주는 모듈; 일종의 interface 역할)**

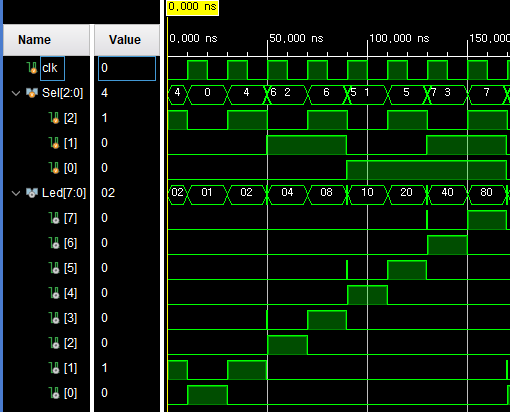


- Mux 모듈과 fpga의 LED와 SWITCH를 연결해 줄 interface 역할을 하는 모듈이다.  
- In1~8을 고정으로 설정한 이유는 경우의 수가 너무 많아서 실험에서는 input data를 한정시키기 위함이다. 그리고 각각 십진수로 0~7을 할당하여 output값으로 어떤 값이 선택됐는지 구분할 수 있게 하였다.  
- Output Led는 fpga의 어느 LED에 불이 켜질지 선택하기 위한 변수다. 즉 mux의 output으로 어떤 값이 선택됐는지 시각적으로 구분하기 위해 설정한 변수다.  
- Led =8’d1은 Led[0]에 1이 담기고 나머지는 0이 담기며, Led=8’d128은 Led[7]에 1이 담기고 나머지는 0이 담긴다.

**3. simulation module**

  
Sel은 selection비트를, Led[0]~[7]는 각각 mux8to1모듈의 In1~8과 대응된다.

**-simulation**

  
**In1~8을 각각 Led[0]~[7]과 대응시키면 된다. 즉 Led[0]이 1이면 In1을 output으로 선택했다는 뜻이다.**

**< truth table>**

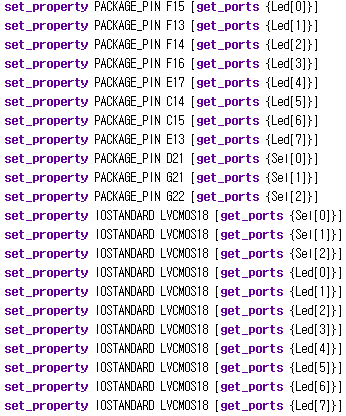
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **output** | **LED** |
| **Sel[0]** | **Sel[1]** | **Sel[2]** | **Out** | **Led** |
| **0** | **0** | **0** | **In1** | **Led[0]** |
| **0** | **0** | **1** | **In2** | **Led[1]** |
| **0** | **1** | **0** | **In3** | **Led[2]** |
| **0** | **1** | **1** | **In4** | **Led[3]** |
| **1** | **0** | **0** | **In5** | **Led[4]** |
| **1** | **0** | **1** | **In6** | **Led[5]** |
| **1** | **1** | **0** | **In7** | **Led[6]** |
| **1** | **1** | **1** | **In8** | **Led[7]** |

**Sel이 000이면 In1이 output으로 선택됐다는 뜻이며 Sel이 101이면 In6이 output으로 선택됐다는 뜻이다. (In1~8은 각각 Led[0]~[7]과 대응)**

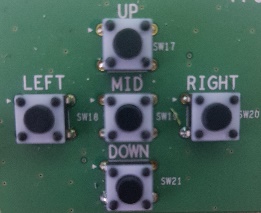
**- FPGA**

업로드:

Design Sources경로에 code 짜기 -> Run Synthesis & Run Implementation -> Constraints경로에 .xdc 파일 추가하기 -> Open Implemented Design 클릭 후 Constraints Wizard 클릭(Define Target, .xdc파일 타겟 설정) -> Window 탭에서 I/O ports 클릭 -> 원하는 pin 선택 및 LVCMOS18(I/O Std) 선택 -> 저장 후 .xdc reload(아래 그림처럼 코드가 자동으로 짜서 나온다.) -> Generate Bitstream 클릭(Synthesis, Implementation도 자동으로 실행) 후에 Open Hardware Manager 클릭 -> Open Target 클릭 후 Auto Connect 클릭 -> Program Device 클릭 -> FPGA에 업로드 끝

<- pin 배정  
Sel[0], [1], [2]는 각각 LEFT MID RIGHT SWITCH에, Led[0]~[8]은 각각 LD1~8에 할당했다.

|  |  |  |
| --- | --- | --- |
| **사진** |  |  |
| **Sel[0] [1] [2]** | **0 0 0** | **0 0 1** |
| **PUSH**  **SWITCH** | **NONE** | **RIGHT** |
| **사진** |  |  |
| **Sel[0] [1] [2]** | **0 1 0** | **0 1 1** |
| **PUSH**  **SWITCH** | **MID** | **MID**  **RIGHT** |
| **사진** |  |  |
| **Sel[0] [1] [2]** | **1 0 0** | **1 0 1** |
| **PUSH**  **SWITCH** | **LEFT** | **LEFT**  **RIGHT** |
| **사진** |  |  |
| **Sel[0] [1] [2]** | **1 1 0** | **1 1 1** |
| **PUSH**  **SWITCH** | **LEFT**  **MID** | **LEFT**  **MID RIGHT** |

 LEFT = input Sel[0], MID = input Sel[1], RIGHT = input Sel[2]

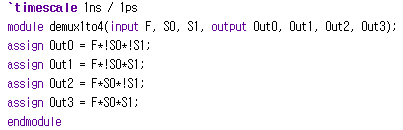
위 표에서 PUSH SWITCH의 switch를 누르면 해당되는 사진처럼 LED가 켜진다.

(LD1~8 = Led[0]~Led[7])

1. 4X16 Decoder(by 1X4 DEMUX)

**-code**

**1. 1X4 DEMUX 모듈**



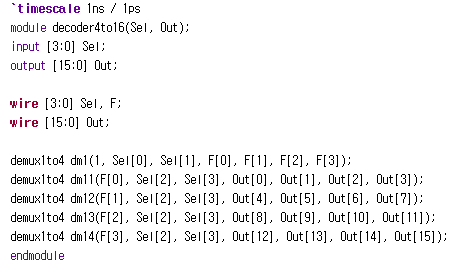
**S0**

**S1**

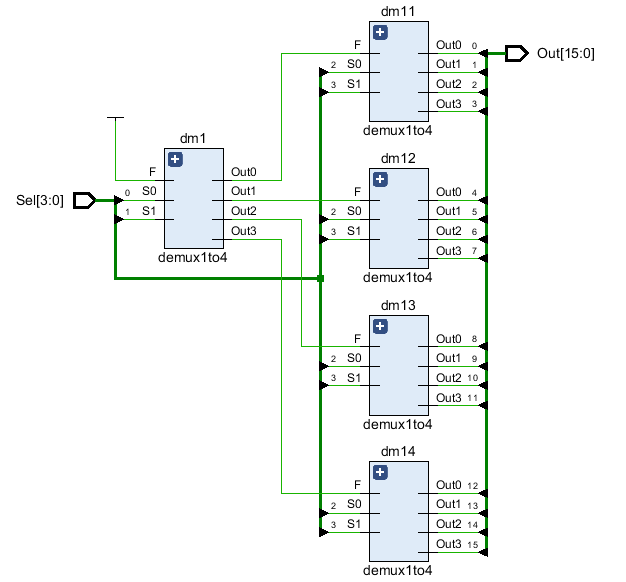
|  |  |
| --- | --- |
| |  | | --- | | 1X4  DEMUX |   **F**  Out0  Out1  Out2  Out3 |

**2. 4X16 Decoder 모듈**

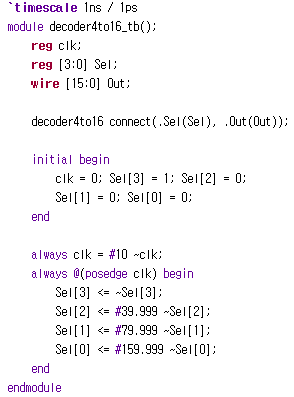
|  |
| --- |
| 4X16  decoder |

  
S0 S1 S2 S3 input 4개와 Out[0] ~ [15] output 16개의 4X16 decoder에 대한 코드

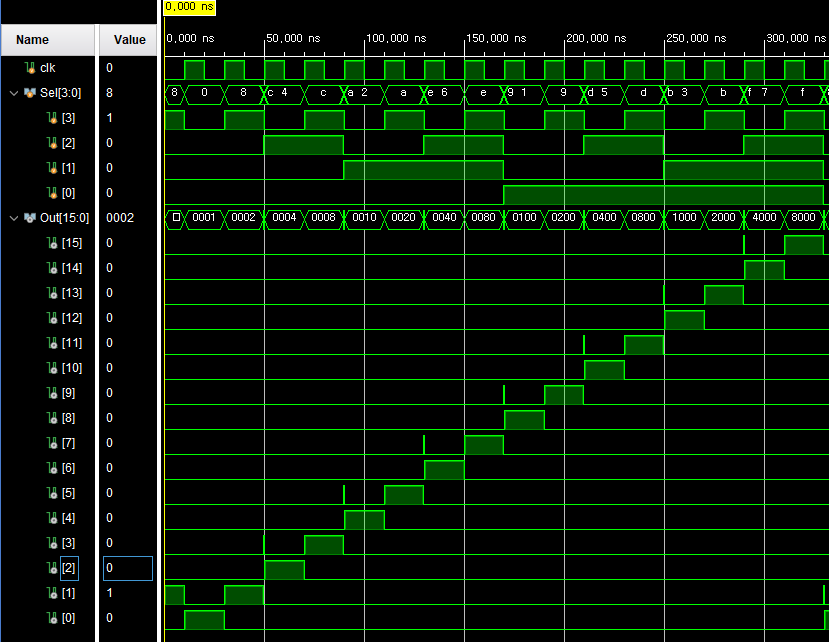
S0~3 ---- Out[0]~[15]



**3. 4to16 decoder simulation 모듈**

****

**-simulation**



**-truth table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | | **Output** |
| **Sel[0]** | **Sel[1]** | **Sel[2]** | **Sel[3]** | **Out** |
| **0** | **0** | **0** | **0** | **Out[0]** |
| **0** | **0** | **0** | **1** | **Out[1]** |
| **0** | **0** | **1** | **0** | **Out[2]** |
| **0** | **0** | **1** | **1** | **Out[3]** |
| **0** | **1** | **0** | **0** | **Out[4]** |
| **0** | **1** | **0** | **1** | **Out[5]** |
| **0** | **1** | **1** | **0** | **Out[6]** |
| **0** | **1** | **1** | **1** | **Out[7]** |
| **1** | **0** | **0** | **0** | **Out[8]** |
| **1** | **0** | **0** | **1** | **Out[9]** |
| **1** | **0** | **1** | **0** | **Out[10]** |
| **1** | **0** | **1** | **1** | **Out[11]** |
| **1** | **1** | **0** | **0** | **Out[12]** |
| **1** | **1** | **0** | **1** | **Out[13]** |
| **1** | **1** | **1** | **0** | **Out[14]** |
| **1** | **1** | **1** | **1** | **Out[15]** |

<- Input Sel[0]~[3]값에 따라 Output은 Out[N]=1, 나머지 Out[0], [1], … [N-1], [N+1], … [15]=0이 된다.

1. 기타 논의 및 검토 사항

처음 8X1 MUX를 fpga로 구현할 때, input을 어떻게 받아야 할지 굉장히 많이 고민했다. 8개의 데이터와 selection bit 3개를 모두 input으로 놓자니 fpga에는 input용 인터페이스가 switch 5개 밖에 없어서 애매한 상황이었다. 결국8X1 MUX 모듈과 fpga를 중간에서 연결해 줄 interface 모듈을 구현함으로써 이를 해결할 수 있었다. 8X1 MUX 모듈에서는 input을 총 11(=8+3)개 output을 8개 구현해 놓고 interface 모듈에서 8X1 MUX 모듈에 8개의 input data를 확정지어서 넣는 방식으로 fpga에서는 총 3개의 input 인터페이스만이 필요하게 되었다.

4X16 decoder를 1X4 demux로 만들기 위해 input개수 4개와 적은 개수의 demux에만 집착하여 총 4개의 demux만으로 decoder를 구현하고자 하다가 굉장히 많은 시간을 낭비했다. 4개로 안 된다면 층을 나누면 어떨까 하는 생각을 하게 되었고 결국 input값 4개를 2개씩 2묶음으로 1층의 demux와 2층의 demux에 각각 한 묶음씩 selection bit로 할당하고 나니 쉽게 풀렸다. 너무 효율만 따지려고 하다가 많은 시간을 소비하게 된 격이다. 앞으로는 효율을 맞춰서 구현하기 보다는 구현을 하고 그에 맞춰서 효율을 살려 보는 방식으로 생각을 해보아도 좋을 듯싶다.

1. 추가 이론

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | | | | | | | | | **Output** | | | |
| **D0** | **D1** | **D2** | **D3** | **D4** | **D5** | **D6** | **D7** | **D8** | **D9** | **A** | **B** | **C** | **D** |
| **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** | **1** |
| **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **1** | **0** |
| **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **0** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0** | **0** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |

위 표는 Decimal to BCD Encoder의 input 과 output을 나타낸 표다.

A = D9’ + D8’  
B = D7’ + D6’ + D5’ + D4’  
C = D7’ + D6’ + D3’ + D2’  
D = D9’ + D7’ + D5’ + D3’ + D1’

각 output은 위 식과 같이 표현할 수 있다. 위 논리식의 의미는 다음과 같다. A의 경우 십진수9와 8을 2진수로 바꿀 때 A의 자리에 1이 들어오게 되고 나머지의 십진수는 0이 온다. B의 경우는 십진수 7, 6, 5, 4의 경우 B의 자리에 1이 오게 되고 나머지의 경우 0이 온다. 마찬가지로 C, D의 자리에는 각각 (7, 6, 3, 2) 와 홀수의 십진수 값을 이진수로 바꿀 때 1이 온다.

|  |  |
| --- | --- |
| Decimal | ABCD |
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |

A의 자리에 1이 오는 경우는 8, 9  
B의 자리에 1이 오는 경우는 4, 5, 6, 7  
C의 자리에 1이 오는 경우는 2, 3, 6, 7  
D의 자리에 1이 오는 경우는 홀수인 것을 볼 수 있다.