

|  |  |
| --- | --- |
|  |  |
| **R E P O R T** | |
|  |  |

제목: 논리 회로 및 실험 8주차 과제

과목: 논리 회로 및 실험

날짜(년/월/일): 2018/05/09

소속 학과: 컴퓨터 전자 시스템 공학부

학번: 201702234

이름: 유동혁

|  |
| --- |
| **본 보고서의 내용 중 다른 문서(자료)를 인용한 것이 있습니까?**  **예( V ) 아니오( )** |
| **위에서 ‘예’로 답한 경우, 인용한 다른 문서는 무엇인지 아래에**  **명시해 주세요. (여러 개의 경우 주요 자료 2개 까지)**  **- 참조:**  [**http://woodforest.tistory.com/123**](http://woodforest.tistory.com/123) |

**논리회로 및 실험 예비레포트**

**학번: 201702234**

**이름: 유동혁**

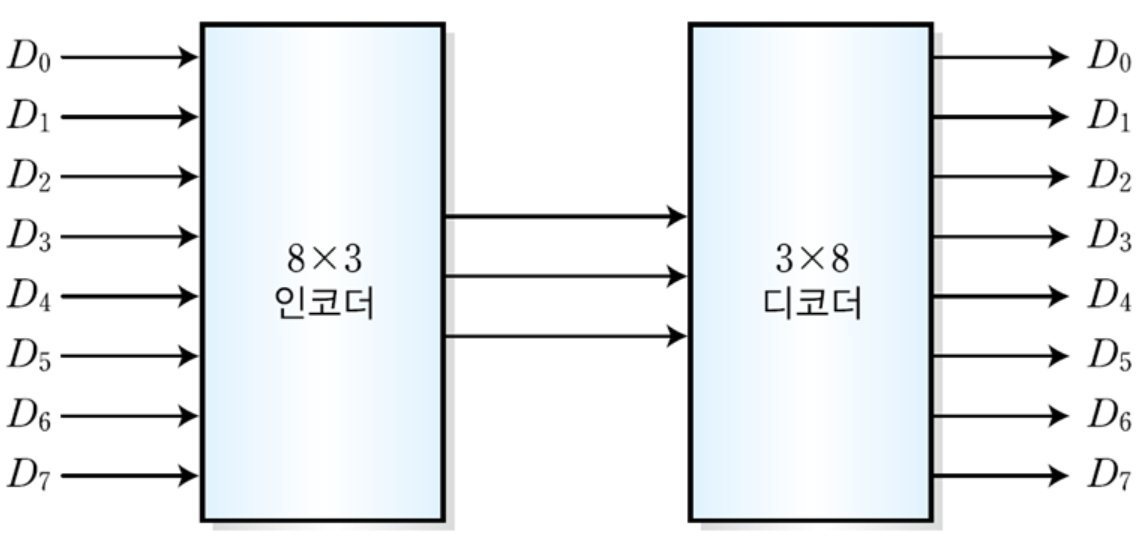
1. 목표: 디코더와 인코더에 대해 조사한다.

2. 내용:

|  |  |
| --- | --- |
| 디코더 (Decoder) | N개의 입력을 받아 2^n개의 출력으로 해독.  복호화 작업을 수행한다.  🡺각 입력에 따라 D중 하나만 1을 출력하고, 다른 D들은 모두 0을 출력한다. |
|  |  |
|  | 3\*8디코더의 회로도와 진리표 |

|  |  |
| --- | --- |
| 인코더 (Encoder) | 2^n개의 입력을 받아 n개의 출력으로 암호화.  부호화 작업을 수행한다.(=디코더와 반대)  각 입력마다 출력되는 값이 다르다. |
| 4\*2 인코더의 회로도와 진리표 |  |
| 8\*3 인코더의 회로도와 진리표 |  |

**인코더와 디코더의 관계도** 부호화 기능을 하는 인코더와 복호화 기능을 하는 디코더가 서로 정반대의 기능을 한다는 것을 알 수 있다.



**논리 회로 및 실험 결과레포트**

**학번: 201702234**

**이름: 유동혁**

1. 실습목표: 반가산기와 전가산기를 이용해 4Bit 가산기를 구현한다.

2. 실습내용:

|  |
| --- |
| \_8thPr\_4bifFull.v |
| module \_8thPr\_4bifFull(  a, b, s, c\_out  );  input [3:0]a, b; //input = a[0], a[1], a[2], a[3] ,, b[0], b[1], b[2], b[3]  output [3:0]s; //S\_output = s[0], s[1], s[2], s[3] == 2^0, 2^1, 2^2, 2^3's output\_value  output c\_out; //C\_Final\_output == 2^4's output\_value  wire [0:2]c; // UNIT2, UNIT3, UNIT4's C\_in value  FULL\_ADDER UNIT1(.X(a[0]), .Y(b[0]), .Z(1'b0), .S(s[0]), .C(c[0]));  FULL\_ADDER UNIT2(.X(a[1]), .Y(b[1]), .Z(c[0]), .S(s[1]), .C(c[1]));  FULL\_ADDER UNIT3(.X(a[2]), .Y(b[2]), .Z(c[1]), .S(s[2]), .C(c[2]));  FULL\_ADDER UNIT4(.X(a[3]), .Y(b[3]), .Z(c[2]), .S(s[3]), .C(c\_out));  endmodule |

|  |
| --- |
| FULL\_ADDER.v |
| module FULL\_ADDER(  X, Y, Z,  S, C  );  input X, Y, Z;  output S, C;  wire S, C;  wire REG\_C1, REG\_C2;  wire REG\_SUM;  HALF\_ADDER UNIT1(.A(X), .B(Y), .S(REG\_SUM), .C(REG\_C1)); //REG\_SUM = X XOR Y, REG\_C1 = X AND Y  HALF\_ADDER UNIT2(.A(Z), .B(REG\_SUM), .S(S), .C(REG\_C2)); //S = Z XOR REG\_SUM, REG\_C2 = Z AND REG\_SUM  //assign S using HALF\_ADDER module  assign C = REG\_C1 | REG\_C2; //C = REG\_C1 OR REG\_C2  //S = Z XOR (X XOR Y), C = (X AND Y) OR (Z AND (X XOR Y))  //C ==> next Full Adder's Z  endmodule |

|  |
| --- |
| HALF\_ADDER.v |
| module HALF\_ADDER(  A, B, S, C  );  input A, B;  output S, C;  assign S = A^B; //S = A XOR B  assign C = A&B; //C = A AND B  endmodule  /\*  HALF\_ADDER unit1 (.A(x), .B(y), .S(sout), .C(cout));  //in C language...  int s = xor(a, b);  int c = and(a, b);  //can return 2 arguments...  int s, c = HALF\_ADDER(a, b)  \*/ |

|  |
| --- |
| HALF\_ADDER의 회로도 |
|  |
| FULL\_ADDER의 회로도 |
|  |
| \_8thPr\_4bifFull의 회로도 |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 진리표 | | | | | | | | | | | | |
| 입력 | | | | | | | | 출력 | | | | |
| A | | | | B | | | |
| A3 | A2 | A1 | A0 | B3 | B2 | B1 | B0 | C\_out | S3 | S2 | S1 | S0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| … | | | | … | | | | … | | | | |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| … | | | | … | | | | … | | | | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

3. 실습결과:

|  |  |
| --- | --- |
| 1111 + 1111 = 11110 | 0101 + 0110 = 01011 |
|  |  |
| 1000 + 1000 = 10000 | 0010 + 1010 = 01100 |
|  |  |

4. 고찰: 파일과 그 안에 있는 함수를 호출하는 방법을 알 수 있었다.