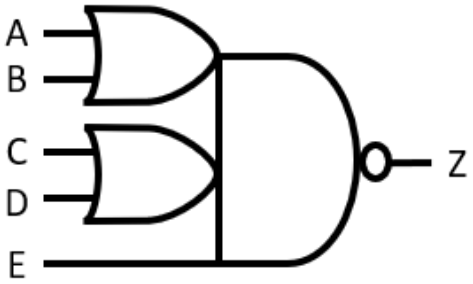
Problem Set 1 Answers - CMOS transistor-level network and hazard circuit

Problem 1

- OAI221 (OR-AND-INVERTER) 을 CMOS transistor-level diagram을 그려 보아라. (※ 중요: P-FET과 N-FET network을 그리는 절차를 보이기 바람). 또한, 총 사용한 transistor는 모두 몇 개인가? (최소 개의 transistor을 사용해야함)



Gate symbol of OR-AND-INV (OAI221) gate

정답 :

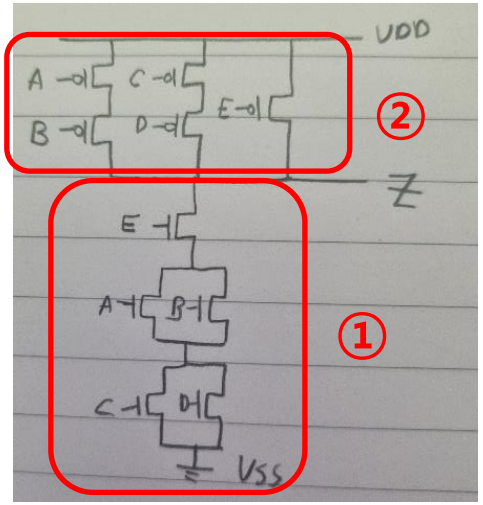
$\bar{f}$에 대한 N-FET network을 먼저 만든다.

$$\mathbf{f_n = \bar{f} = (A \vee B) \wedge (C \vee D) \wedge E}$$

f 의 duality를 사용해서 P-FET network를 만든다.

$$\mathbf{f = \overline{(A \vee B) \wedge (C \vee D) \wedge E} = (\bar{A} \wedge \bar{B}) \vee (\bar{C} \wedge \bar{D}) \vee \bar{E}}$$

따라서, transistor 개수는 10개이다.



Problem 2

- 각각 1-bit인 4개의 inputs a, b, x, y와 1개의 1-bit인 output z로 이루어진 회로가 있다. 4개의 data input들은 아래의 테이블과 같은 logic operation대로 회로가 동작한다. 다음 질문들에 대하여 상세히 답하시오.

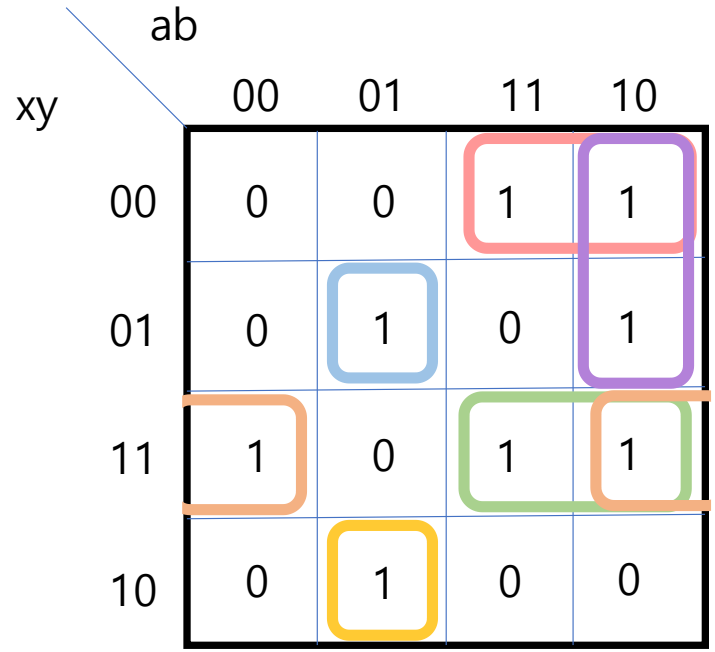
a	b	z
0	0	$x \wedge y$
0	1	$(\bar{x} \wedge y) \vee (x \wedge \bar{y})$
1	0	$\bar{x} \vee y$
1	1	$x == y$

Gate	Delay(ns)
2-input OR	1
2-input AND	1
3-input OR	1
INV	2

Problem 2.1

- 회로의 동작을 표현하는 Karnaugh-map을 그리시오.

정답 :



Problem 2.2

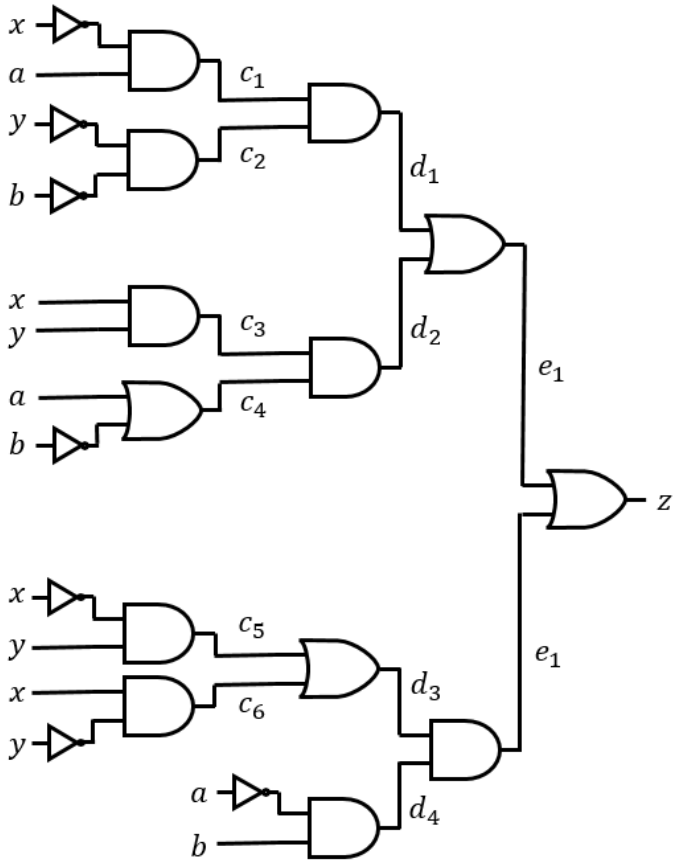
- 위의 회로에서 Hazard 있는 회로를 설계한다고 가정하자. 2-input OR, 2-input AND, INV만을 이용해 4-level 회로를 그리고 그에 해당하는 simplified Sum-of-products (SOP)으로 표현하시오. 또한 delay표를 참고하여 시간에 따른 signal graph를 통해 Hazard가 언제 발생하는지 그리시오. (단, Hazard와 관련된 signal은 1ns에 변한다고 가정하고 inverter는 level 계산에 포함되지 않는다고 한다.)

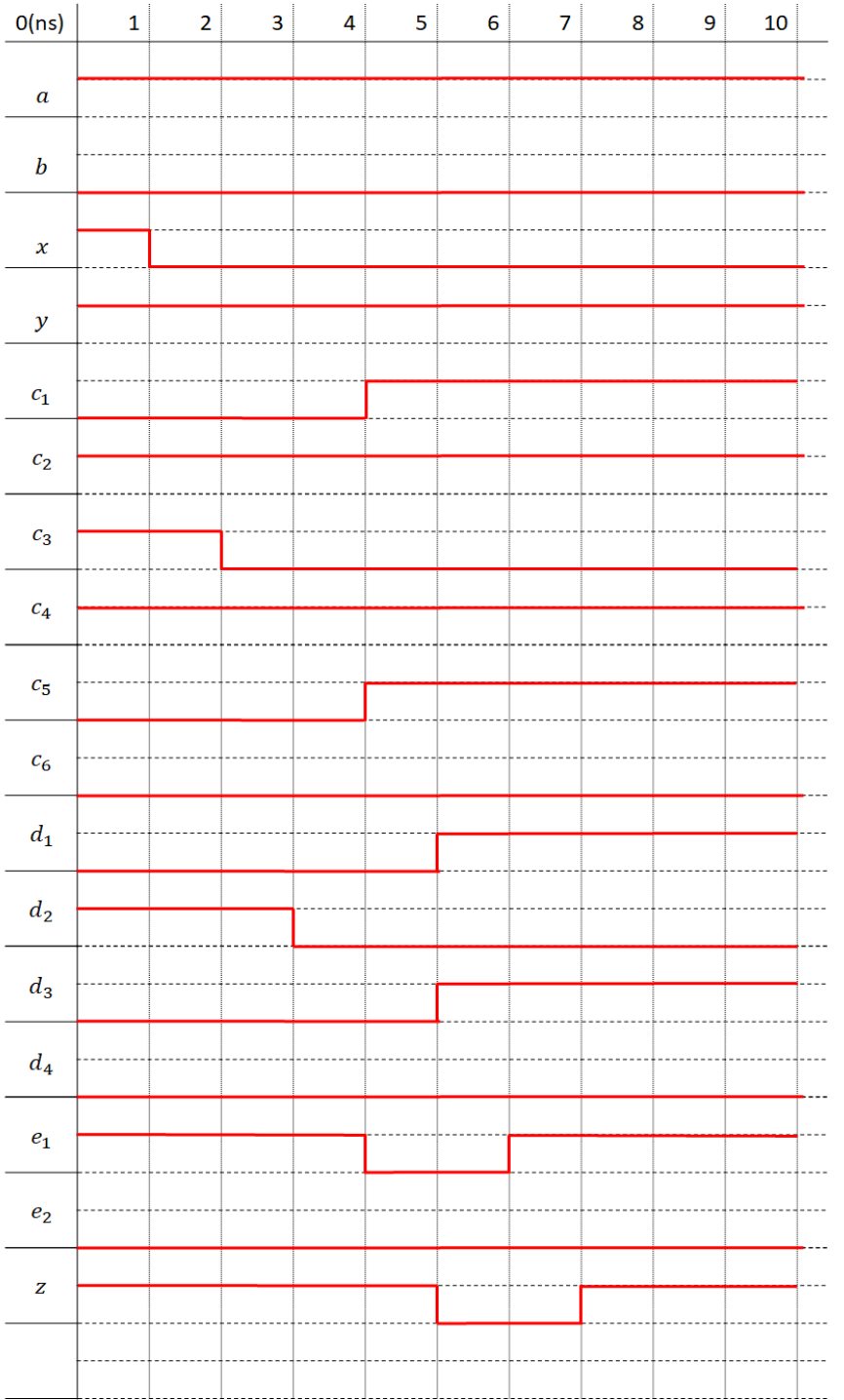
SOP :

$$(\overline{x} \wedge \overline{y} \wedge a) \vee \left(\overline{x} \wedge a \wedge \overline{b}\right) \vee (x \wedge y \wedge a) \vee \left(x \wedge y \wedge \overline{b}\right) \vee (\overline{x} \wedge y \wedge \overline{a} \wedge b) \vee (x \wedge \overline{y} \wedge \overline{a} \wedge b)$$

$$= ((\overline{x} \wedge a) \wedge (\overline{y} \vee \overline{b})) \vee ((x \wedge y) \wedge (a \vee \overline{b})) \vee ((\overline{a} \wedge b) \wedge ((\overline{x} \wedge y) \vee (x \wedge \overline{y})))$$

회로 :



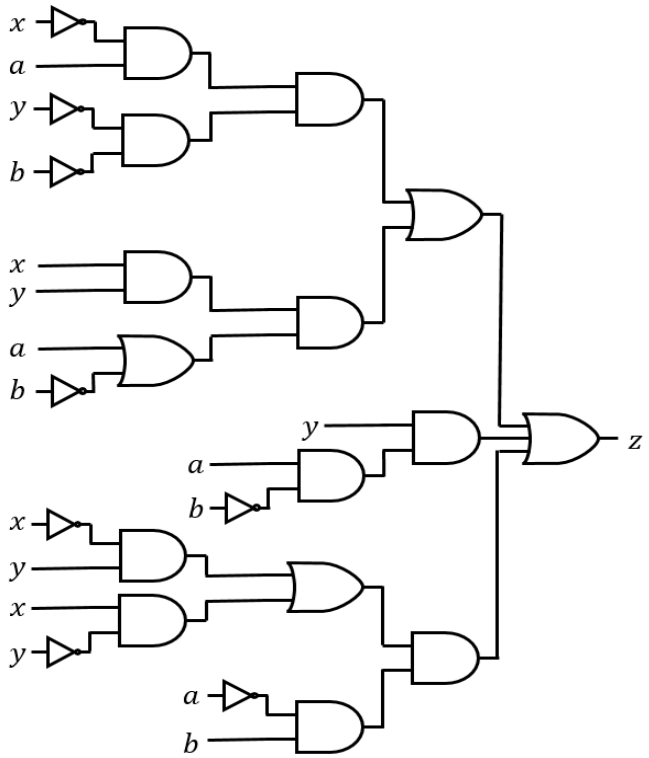


Problem 2.3

- 3-input OR, 2-input OR, 2-input AND gate, INV만을 사용하여 2.2 에서 발생한 Hazard를 피하도록 4-level 회로를 그리시오.

정답 예시 : $(y \wedge a \wedge \overline{b})$**를 기존 problem 2- 2의 식에 추가**

$$(\overline{x} \wedge \overline{y} \wedge a) \vee \left(\overline{x} \wedge a \wedge \overline{b}\right) \vee (x \wedge y \wedge a) \vee \left(x \wedge y \wedge \overline{b}\right) \vee (\overline{x} \wedge y \wedge \overline{a} \wedge b) \vee (x \wedge \overline{y} \wedge \overline{a} \wedge b) \vee (y \wedge a \wedge \overline{b})$$



Problem Set 2 answers – Boolean logic optimization and combinational logic

Problem 1

- F = (a & b & d) | (~a & c) | (b & c & d)를 Boolean axioms과 theorems을 적용해서 F = (a & b & d) | (~a & c) 로 간략히 됨을 보여라.

정답:

F = (a & b & d) | (~a & c) | (b & c & d)

= (a & b & d) | (~a & c) (∵ Consensus theorem)

Problem 2

- 아래 Verilog code는 입력 r[5:0]에 대해 c[5:0]을 구하는 combinational logic을 만들기 위한 것으로 작성한 Verilog code이다. 실제로는 combinational logic이 맞는가? 만일 아니라면, gate-level diagram을 그려 왜 combinational logic이 아닌지 지적해 보아라.
  - wire [5:0] c = ({~r[4:0], ~r[5]} & {c[4:0], c[5]}) | 6‘b000100 ;

정답: Combinational logic이다.

이유 : 식을 정리하면 다음과 같이 combinational logic으로 표현 가능

c[2] = 1

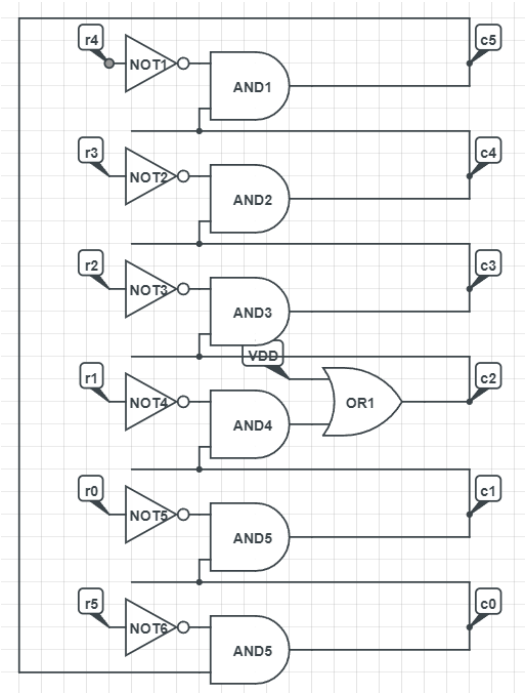
c[3] = ~r[2]

c[4] = ~r[3] ∧ ~r[2]

c[5] = ~r[4] ∧ ~r[3] ∧ ~r[2]

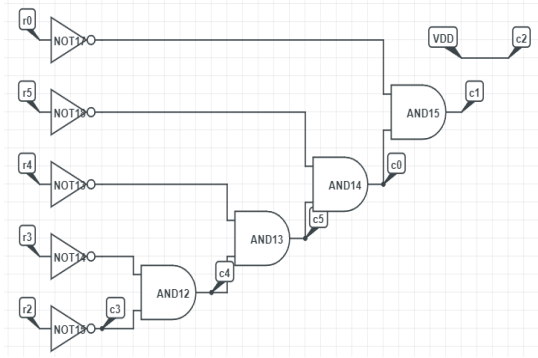
c[0] = ~r[5] ∧ ~r[4] ∧ ~r[3] ∧ ~r[2]

c[1] = ~r[0] ∧ ~r[5] ∧ ~r[4] ∧ ~r[3] ∧ ~r[2]



(오른쪽 그림의 OR1의 두 번째 input에 상관없이 c[2]가 1임에 유의,

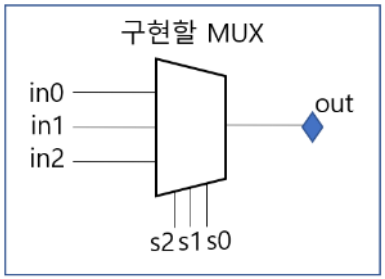
결과적으로 아래 그림과 같은 acyclic한 diagram이 된다.)

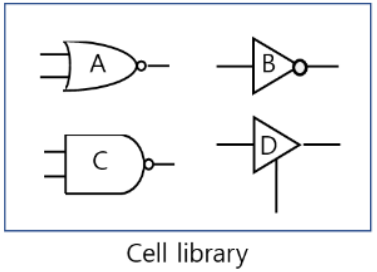


Problem Set 3 Answers – MUX design

Problem 1.1

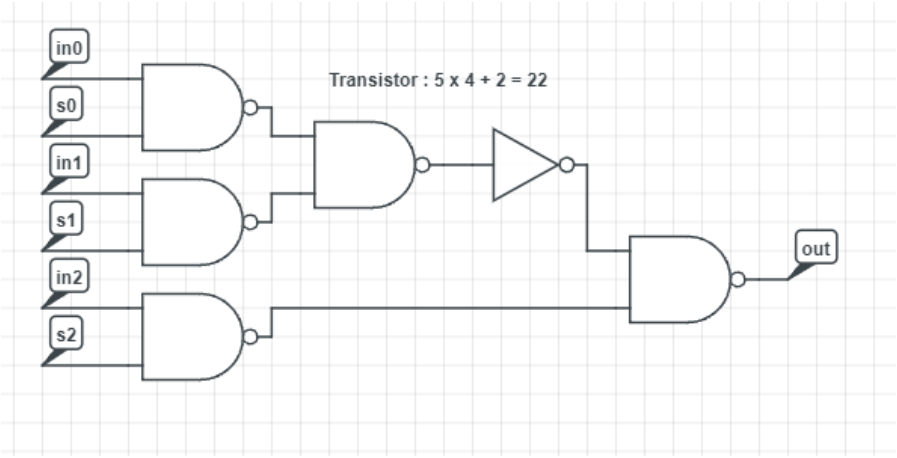
- 1-hot select를 가지는 MUX를 설계하려고 한다. 다음 물음에 답하시오.
  - 아래의 1-hot select을 가지는 MUX에 대해 cell library의 A, B, C cell type 만을 instantiate 해서 구현한 gate-level schematic을 그려 보아라. 총 사용한 transistor는 모두 몇 개인가? (최소 개의 transistor을 사용해야 함)





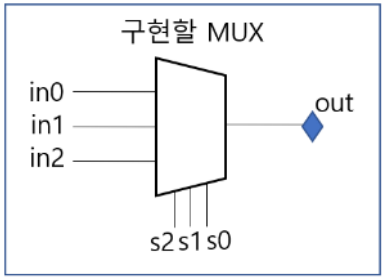
Cell library

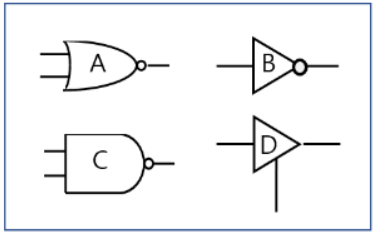
정답 : 아래 그림과 같이 gate를 연결해주면 된다. NAND gate는 transistor 4개, NOT gate는 transistor 2개로 구성되어 있으므로 필요한 transistor의 개수는 4 X 5 + 2 = 22개이다.



Problem 1.2

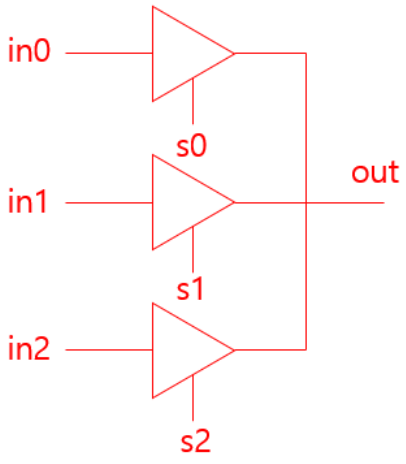
- 1-hot select를 가지는 MUX를 설계하려고 한다. 다음 물음에 답하시오.
  - 이번에는 B, D cell type만을 instantiate 해서 구현한 gate-level schematic을 그려 보아라. 총 사용한 transistor는 모두 몇 개인가? (최소 개의 transistor을 사용해야 함)

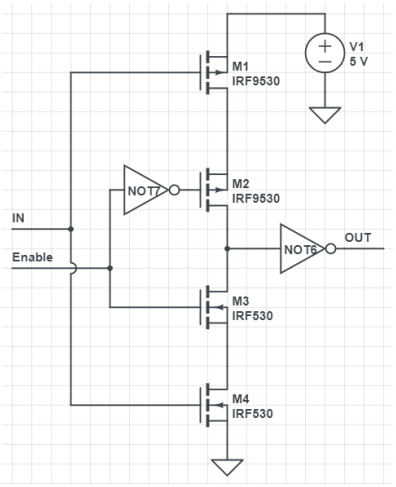




Cell library

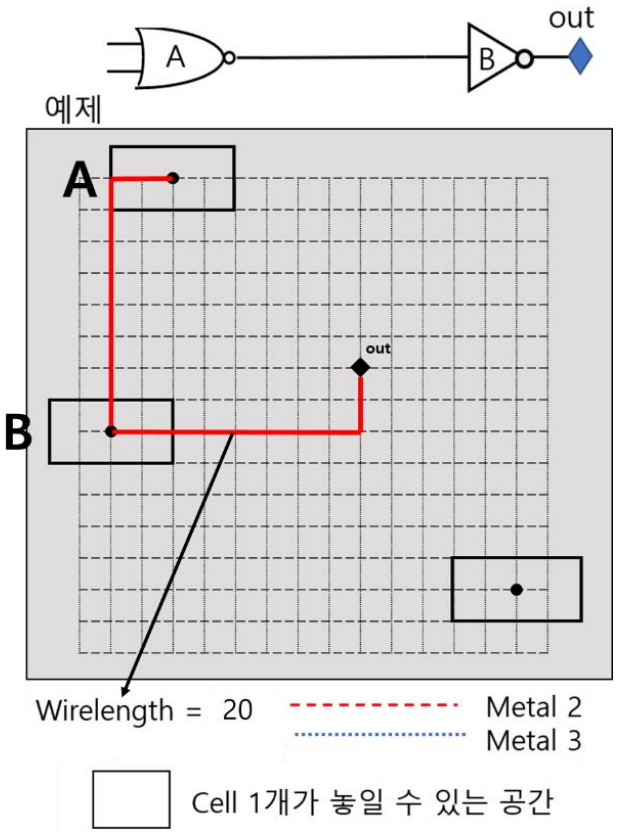
정답 : 아래 왼쪽 그림과 같이 tri-state gate 3개를 연결해 이어주면 되며, tri-state gate 구현을 위해서는 오른쪽 그림과 같이 총 4개 + 2개(inverter)*2=8개가 필요하므로, multiplexer를 구현하기 위해 필요한 transistor의 개수는 24개이다.





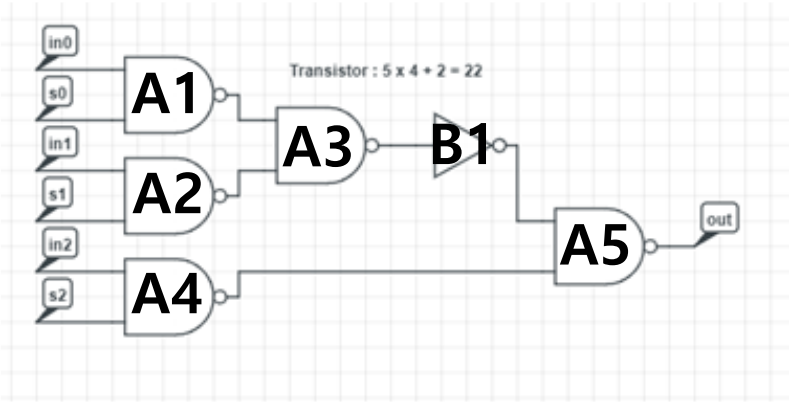
Problem 1.3

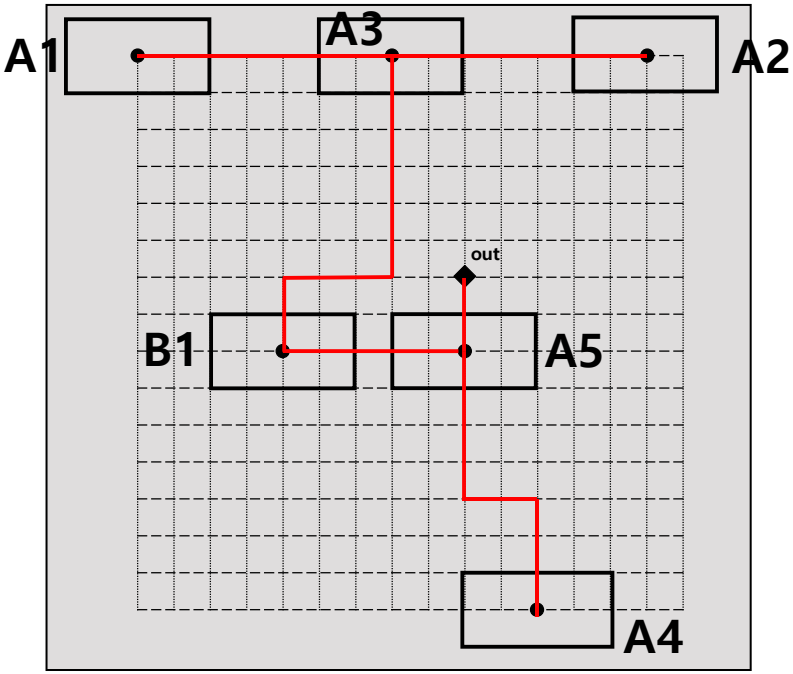
- 아래와 같은 gate-level schematic을 die에 구현(placement & routing) 한 결과는 다음과 같다.



Problem 1.3

- 이 때, 1.1에서 구한 gate-level schematic을 아래의 die에 최소의 wire length로 구현(placement & routing)해 보아라. 이 때 구현한 wire length는 얼마인가? (단, 1.1)의 답안에서 어떤 gate가 사용되었는지 명시할 것)

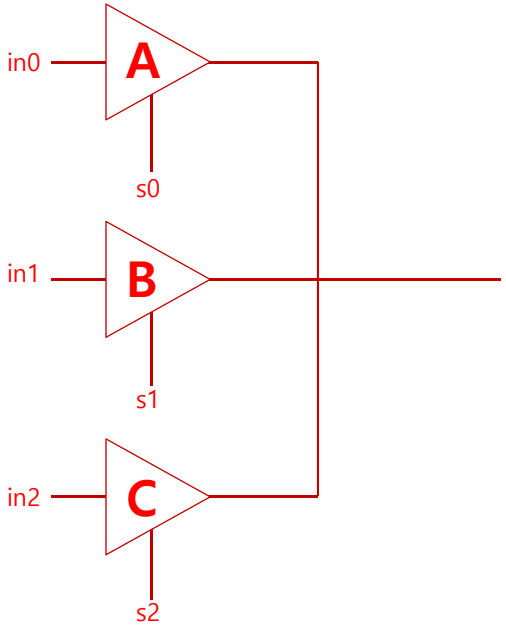


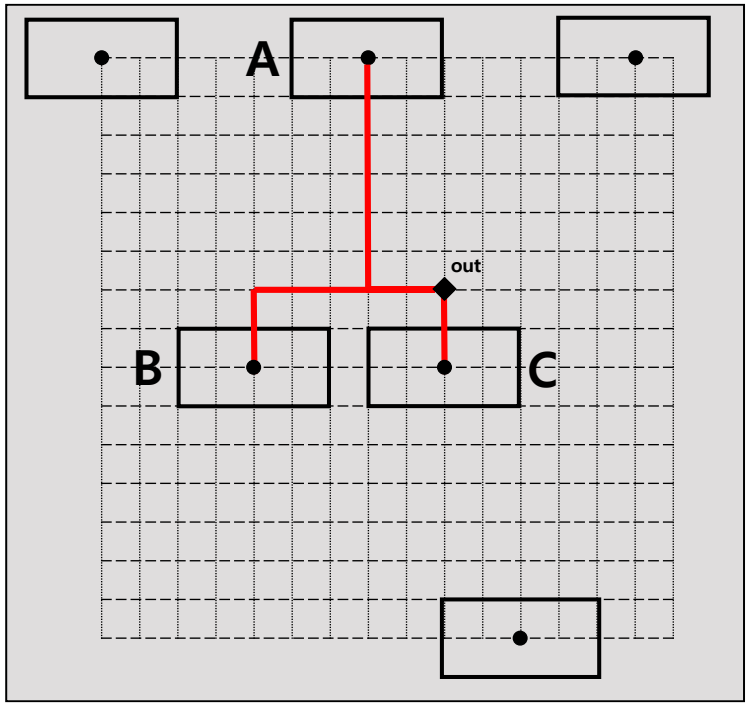


Wire length : 41

Problem 1.4

- 이번에는 1.2에서 구한 gate-level schematic을 아래의 die에 최소의 wire length로 구현(placement & routing)해 보아라. 이 때 구현한 wire length는 얼마인가? (단, 1.2)의 답안에서 어떤 gate가 사용되었는지 명시할 것)





Wire length : 15

Problem 1.5

- 문제 1.1 ~ 1.4까지의 해결을 통해 살펴본 바, 결론적으로 {A, B, C} 와 {B, D} 사용의 MUX 구현 방식에서의 상대적인 크게 구별되는 장단점은 어떤 것이 있다고 보는가?

정답:

{A, B, C} 장점:

1. Floating state output이 생기지 않음. 즉, 만일 Tri-state 기반의 작은 MUX 를 사용해서 큰 MUX를 만들 때 (예: 6-inpit MUX를 3-input MUX 2개와 2-input OR gate 1개로 구현할 때) 작은 MUX의 floating output을 연결해서 큰 MUX를 구현하는데 어려움이 있음. 하지만 CMOS gate 기반 MUX는 이러한 문제에서 자유로움.

2. 사용하는 transistor의 개수가 적음

{A, B, C} 단점: 1. Wirelength를 더 많이 사용해야 함

Problem Set 4 answers – Decoder design

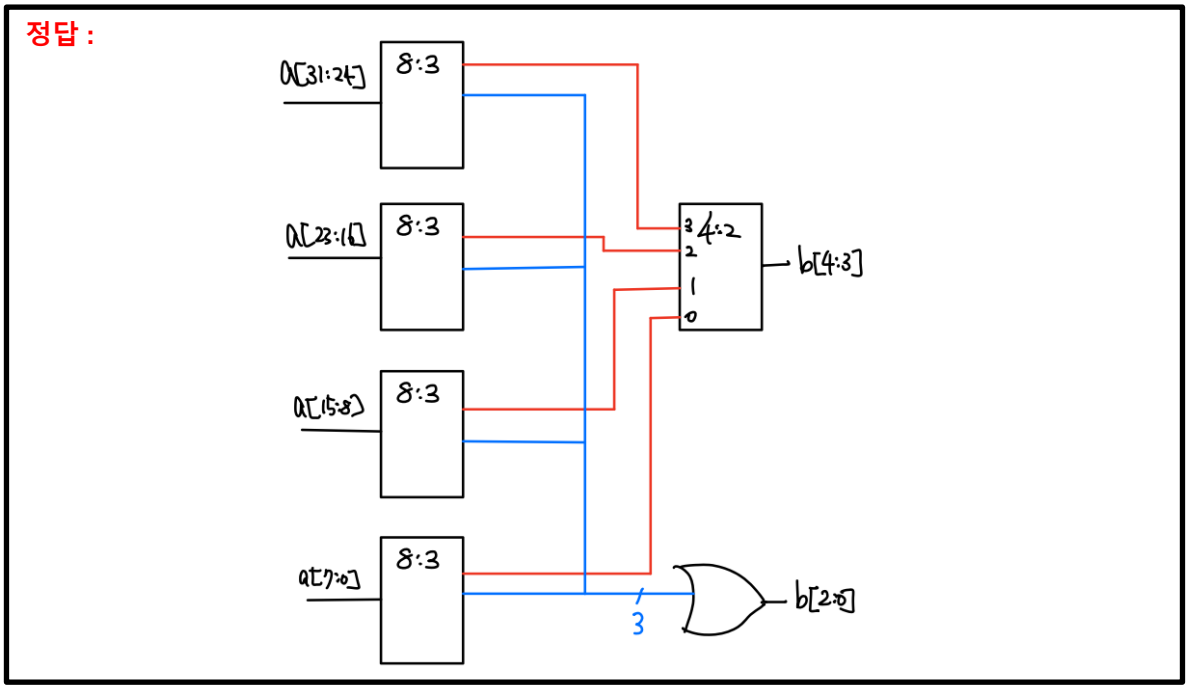
Problem 1

- 수업에서 4:2 encoders를 사용해서 16:4 encoder를 만드는 방법을 학습하였다. 이때 기법으로 4:2 encoders에 “summary” 출력을 추가로 내도록 한 4:2 encoders를 활용함으로써 16:4 encoder 구현의 효율성을 높였다. 다음 소문제에 대해 답하시오.

Problem 1.1

- 4:2 encoder(s)와 8:3 encoder(s)을 사용해서 32:5 encoder (입력 a, 출력 b로 표시)를 구현한 block diagram을 보여라. (구현에 사용하는 encoder(s)에는 summary 출력이 있다고 가정한다.)

정답 :



Problem 1.2

- 문제 1.1에서 구한 block diagram을 활용해서 이제 입/출력 이 바뀐 (즉, 역 방향의 data 흐름) 회로를 만들고자 한다. 구체적으로, 2:4 decoder(s)과 3:8 decoder(s)을 사용하여 5:32 decoder의 block diagram을 구하는 문제이다. 5:32 decoder 구현에 사용할 (4:2 encoder의 summary 와 비슷한 개념이 들어간) 2:4 decoder 또는 3:8 decoder의 Verilog module을 기술해 보아라. (두개 decoder중 하나만 기술하면 됨)
  - 힌트) 이를 위해 문제 1.1에서 사용한 encoder(s)를 decoder(s)로 바꾸는 과정이 필요하다. 또한 encoder(s)에서 사용한 summary 출력을 decoder(s)에서는 어떤 개념으로 바꾸어 출력 또는 입력에 붙일지도 고민해야 한다.

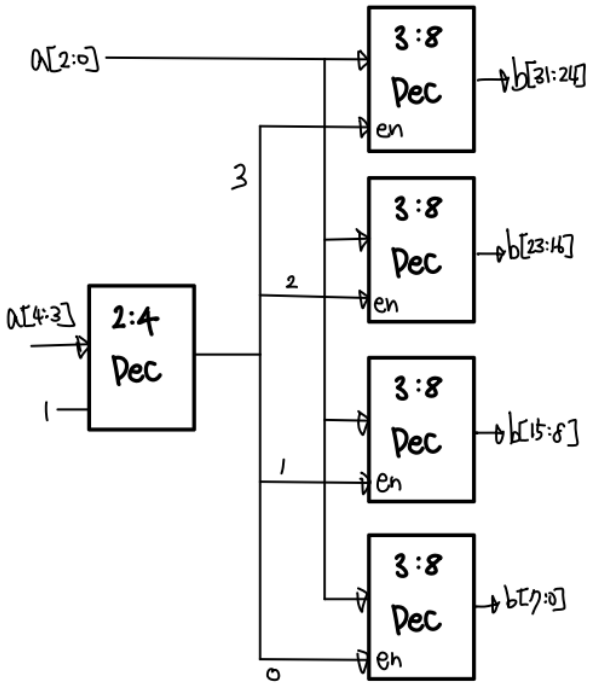
정답 :

```
module Dec2_4 (a,b,en);
    input [1:0] a;
    input en;
    output [3:0] b;
    wire [3:0] c;
    assign c = 1 << a;
    assign b = c & {4{en}};
endmodule
```

Problem 1.3

- 문제 1.2에서 구한 2:4 decoder(s), 3:8 decoder(s)를 활용해서 5:32 decoder (입력 b, 출력 a로 표시)의 block diagram을 완성하여라.

정답 :



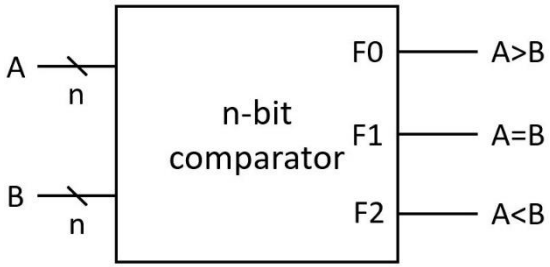
Problem Set 5 Answers – Comparator

Problem 1

- 다음 2-bit comparator **comp2** module을 이용하여 4-bit comparator **comp4** module을 만들려고 한다. 다음 빈칸에 4-bit comparator **comp4** module을 structural description(즉, 내부 모든 module이나 gate는 instantiation으로 표현)으로 작성하여라. 이때, 최소 개수의 2-input **and** module과 2-input **or** module만을 사용해야 한다. (wire는 추가로 선언할 수 있다.)

2-bit comparator

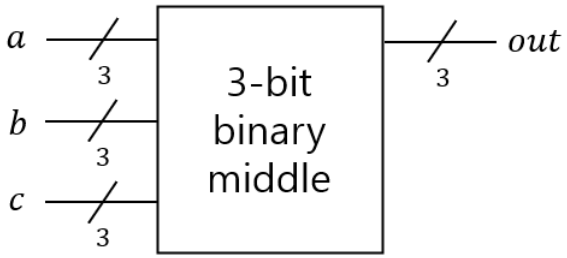
```
module comp2 (a, b, f0, f1, f2);
    parameter n = 2;
    input [n-1:0]          a, b;
    output                 f0, f1,
f2;
    assign f0 = (a > b);
    assign f1 = (a == b);
    assign f2 = (a < b);
endmodule
```



```
module comp4 (A, B, F0, F1, F2);
    parameter n = 4;
    input [n-1:0]         A, B;
    output                F0, F1, F2;

    wire f0, f1, f2, f3, f4, f5;

    comp2 comp2_1 (.a(A[3:2]), .b(B[3:2]), .f0(f0), .f1(f1), .f2(f2));
    comp2 comp2_2 (.a(A[1:0]), .b(B[1:0]), .f0(f3), .f1(f4), .f2(f5));

    wire x0, x1;
    and and1 (x0, f1, f3);
    and and2 (F1, f1, f4);
    and and3 (x1, f1, f5);
    or or1 (F0, f0, x0);
    or or2 (F2, f2, x1);


endmodule
```

Problem 2

- 아래 그림과 같이 3-bit unsigned binary a, b, c가 입력으로 주어졌을 때 세 값 중 중앙값(세 값 중 두 번째로 큰 값, 또는 세 값 중 두 번째로 작은 값)을 출력하는 3-bit binary middle circuit을 구현하고자 한다. 설계의 편의를 위해 a, b, c는 모두 다른 값으로만 입력이 들어온다고 가정하자. 예를 들어, a=110, b=001, c=010일 경우 out=010이다. 이 때 다음 물음에 답하시오.



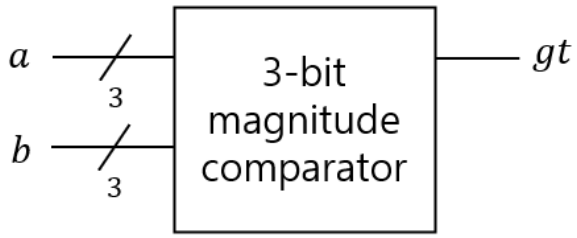
Problem 2.1

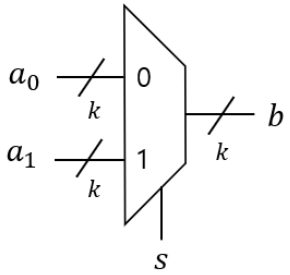
- 3-bit binary middle circuit을 Verilog를 이용하여 구현하려 할 때, 아래 빈 칸을 작성하시오.

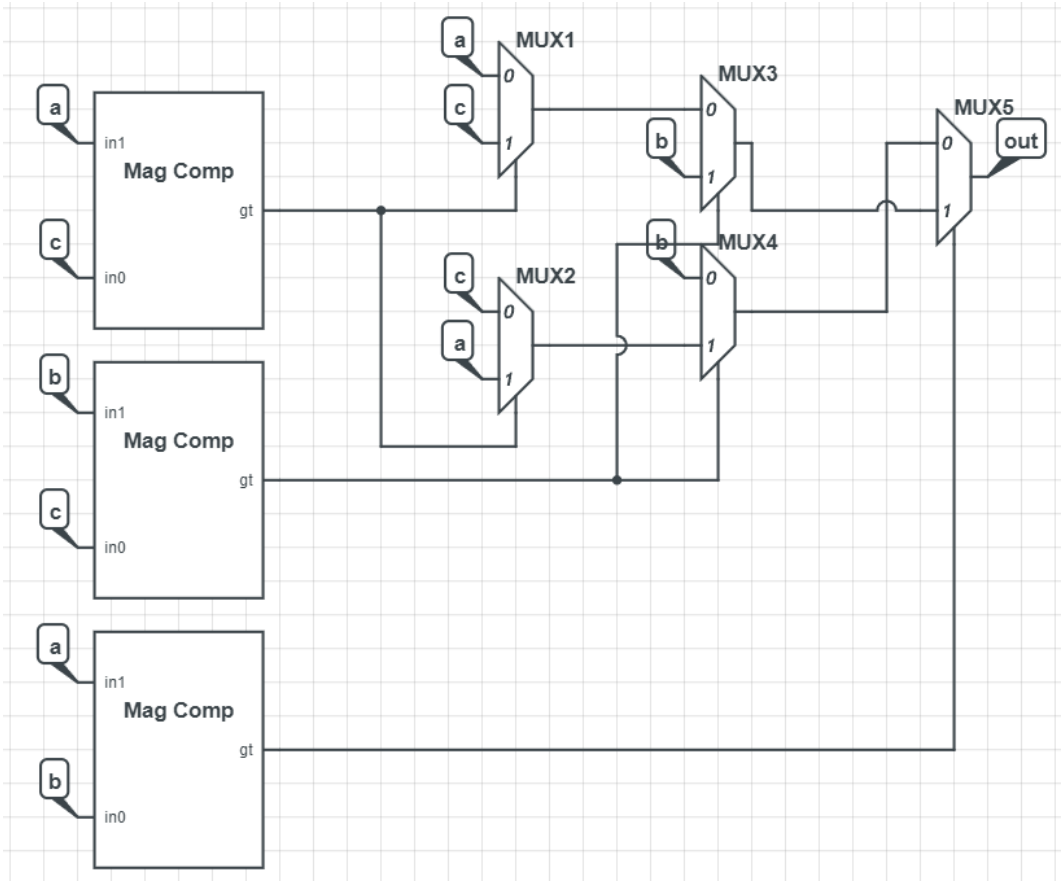
```
module BinaryMiddle(a, b, c, out);
              input [2:0] a, b, c ;
              output reg [2:0] out ;
              always @(*)
              begin
                 out = (a > b) ? ((b > c) ? b : ((a > c) ? c : a)) : ((b > c) ? ((a > c) ? a : c ) : b ) ;
              end
endmodule
```

Problem 2.2

- 아래의 3-bit magnitude comparator, 2:1 multiplexer만을 이용하여 3-bit binary middle circuit의 block diagram을 그리시오. (3-bit magnitude comparator는 두 개의 3-bit unsigned binary a, b를 입력으로 받으며, a > b인 경우 gt=1, 그렇지 않은 경우 gt=0을 출력한다.)






a
MUX1
0
c
1
MUX3
0
b
1
MUX5
0
out
1
a
in1
Mag Comp
gt
c
in0
b
MUX4
c
MUX2
0
a
1
b
in1
Mag Comp
gt
c
in0
a
in1
Mag Comp
gt
b
in0

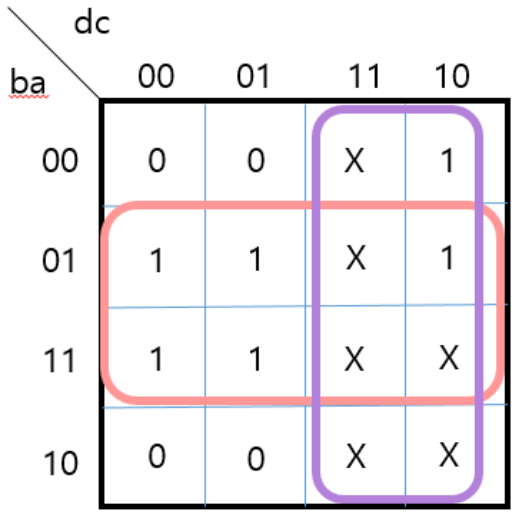
Problem Set 6 Answers – Karnaugh map

Problem 3

- Karnaugh map을 사용해서 아래 수식 f에 대한 simplified Sum-of-products (SOP)를 구해 보아라

$$f = \sum_{dcba} m(1,3,5,7,8,9) + D(10,11,12,13,14,15)$$

정답 :



avd