3장. 조합회로 설계

오늘의 교훈

輕敵必敗

조합회로

- 논리게이트로 구성되며 출력은 입력값으로 결정됨
- 조합회로의 블록다이아그램 (그림 3-1)
- 순서회로
  - 저장요소를 가짐 혹은 feedback 요소를 가짐
  - 출력은 입력과 저장요소의 값으로 결정됨

  $\rightarrow$ 현재의 입력과 과거의 입력에 의존하여 결정됨

설계사항

- 3가지 중요한 설계개념
  - 계층적 설계, 컴퓨터 보조 설계, 하향식 설계
- 계층적 설계
  - VLSI 는 보통 수백만개의 게이트로 구성됨
  - 복잡함으로 한번에 설계할수 없음 →divide and conquer 방법 사용
  - 분할된 작은 조각을 block 이라고 부름.
  - 계층적 설계 →(*그림 3-2, 3-3)
  - 기본 블록 (primitive block)
    * 논리회로의 최소단위, 기호만 사용, schematic 없음
    * ex) NAND 게이트

설계사항 (Cont'd)

- 미리정의된 블록 (predefined block)
  - 기본블록 보다 복잡, 기호 사용, schematic 없음
  - ex) XOR 게이트
- block 의 재사용 →동일한 block 에 대하여 한번만 설계하고 나머지는 재사용
- 기능 블록 (functional block)
  - 미리정의되고 재사용가능하고 디지털 설계에서 널리 사용되는 블록
  - MSI 수준의 설계시 자주 사용됨
  - CAD library 로 지원됨

- 컴퓨터 보조 설계
  - 복잡한 집적회로 설계시 CAD (Computer-Aided Design) 이 없으면 설계가 불가
  - Schematic capture
    - 블록을 그리고 계층별 상호연결을 가능하게 함
    - 기본블록 및 기능블록 수준의 library 가 제공됨

설계사항 (Cont'd)

  - 논리 시뮬레이터 (Logic simulator)
    * 설계된 회로의 검증을 위하여 회로에 시간별로 입력및 출력을 생성
  - 논리 합성 (Logic synthesis)
    * 블록의 기술된 명세서를 이용하여 자동 논리 구현
    * 명세 기술방법
      · 진리표
      · 하드웨어 기술언어 (HDL (Hardware Description Language) 사용)
    * 자동칩 배치와 라우팅 도구에의해 상호연결되고 집적회로 배선까지 구성
    * 설계 HDL 기술에서 집적회로 설계까지 자동적으로 구성할수 있음

- 하향식 (top down) 설계
  - 큰 블록을 작은 블록으로 필요한 만큼 나누어 구현
  - 가끔식 상향식 (bottom up) 설계가 필요할때가 있음

분석 절차

- 조합회로의 분석
  - 회로가 수행하는 기능을 결정
  - 논리도를 이용하여 회로의 기능설명과 함께 부울함수나 진리표를 제시
  - 증명: 기 기술된 설명과 분석된 기능이 일치하는가 검사
  - 분석 방법
    - 부울수식이나 진리표를 수작업이나 논리 시뮬레이션을 사용하여 분석
- 분석 절차
  - 조합회로인가를 확인
    - feedback 이나 기억소자가 없는지 확인
    - 부울함수나 진리표를 구함
    - 회로동작을 해석

분석 절차 (Cont'd)

- 부울함수의 유도
  - 유도 과정
    1. 입력변수 와 보수에대한 출력에 임의의 부호 표시 후 출력함수를 구함
    2. 입력변수와 부호에 대한 출력에 임의의 부호 표시후 출력함수를 구함
    3. 최종출력이 얻어질때까지 단계2의 절차 반복
  - 예) *그림 3-4
- 진리표의 유도
  - 논리도로부터 유도
    - 과정
      1. 입력의 갯수 결정 (n개의 입력에 대해 0에서 $2^n - 1$ 개의 2진수 기입)
      2. 각 블록출력에 임의의 기호를 붙여 하나의 출력을 가진 블록으로 분리
      3. 입력변수만으로 결정되는 블록에 대하여 진리표를 구함
      4. 모든 출력에 대하여 구하여 질때까지 반복
    - 예) *그림 3-5 (이진 가산기) →진리표 (표 3-1)

분석 절차 (Cont'd)

- 논리 시뮬레이션
  - 네트 리스트 (net list) : 시뮬레이터가 읽을수 있는 형태로 기술된 회로
  - Schematic capture tool : 논리도를 그릴수 있게 하는 도구
  - *그림 3-6 : ViewDraw 로 그린 schematic
  - *그림 3-7 : ViewTrace 에 의해 생성된 파형

설계절차

- 설계 : 문제 →논리도나 부울함수
- 과정
  1. 필요한 입출력 갯수를 결정하고 기호할당
  2. 진리표를 작성
  3. 간략화된 부울함수를 얻음
  4. 논리도를 그림
  5. 설계결과과 맞는지 확인
- 예제 3-1

코드 컨버터

- 한 코드를 다른 코드로 변환

- 예) 3-2
  - BCD-to-Excess-3 코드 컨버터
  - 진리표 (*표3-2)
  - 간략화된 부울함수를 얻음 (*그림 3-9)
  - 최종적으로

    $z = D'$

    $y = CD + C'D' = (C \oplus D)'$

    $x = B'C + B'D + BC'D' = B'(C + D)$

    $+ BC'D'$

    $w = A + BC + BD = A + B(C + D)$
  - 논리도를 그림: *그림 3-10

- 예) 3-3
  - BCD-to-7 segment 디코더
  - 7-segment 표시장치

    (*그림 3-11)
  - 진리표 (*표 3-3)
  - 논리간략화 (p. 117)

디코우더 (decoder, 복호기)

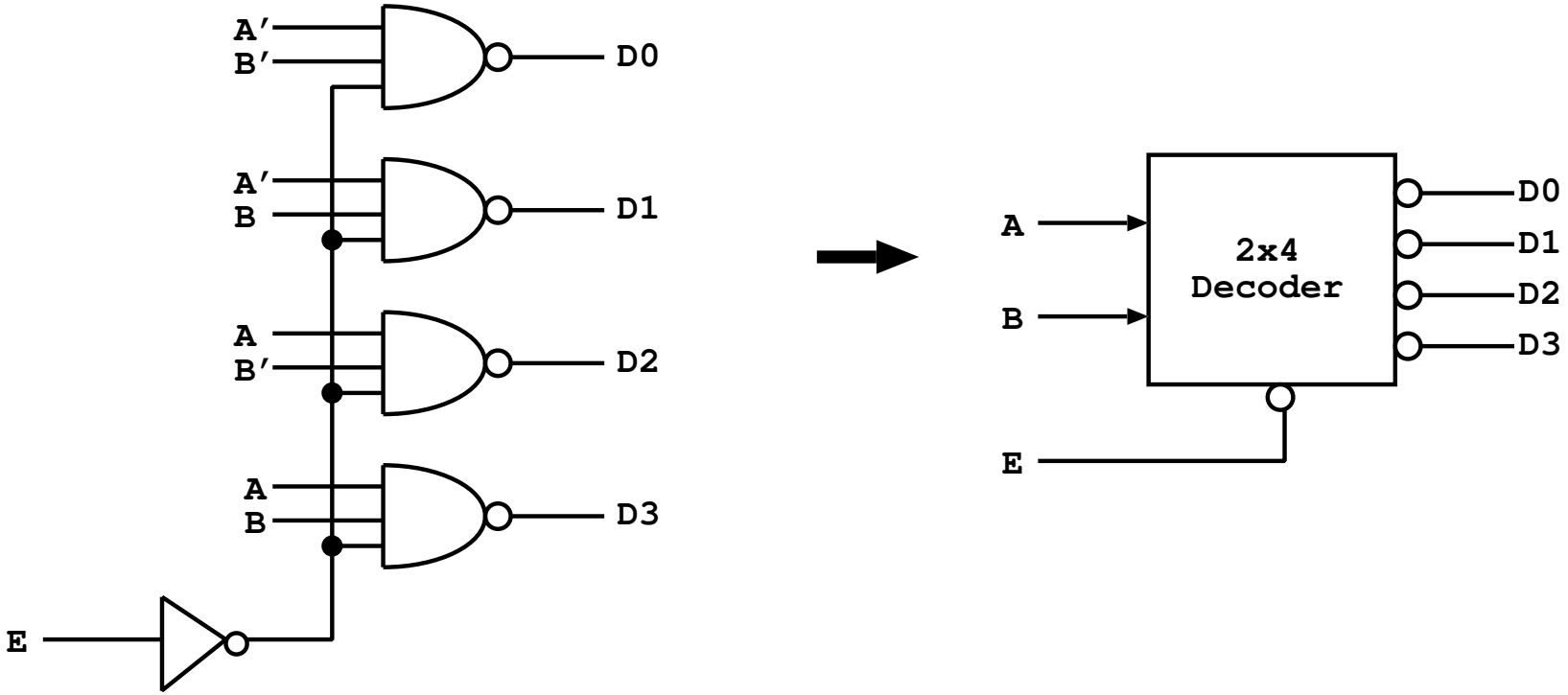
- decoder : code →정보
- encoder : 정보 →code
- n bit 2진 code 시
  - 입력 : n
  - 출력 : 2^n
- 보통 $n \times 2^n$ decoder 라 불림
- 예) 3 × 8 decoder
  - 진리표 : 표3-4

x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

- 논리설계 : 각출력은 민텀항에 해당 (*그림 3-12)

디코우더 (decoder, 복호기) (Cont'd)

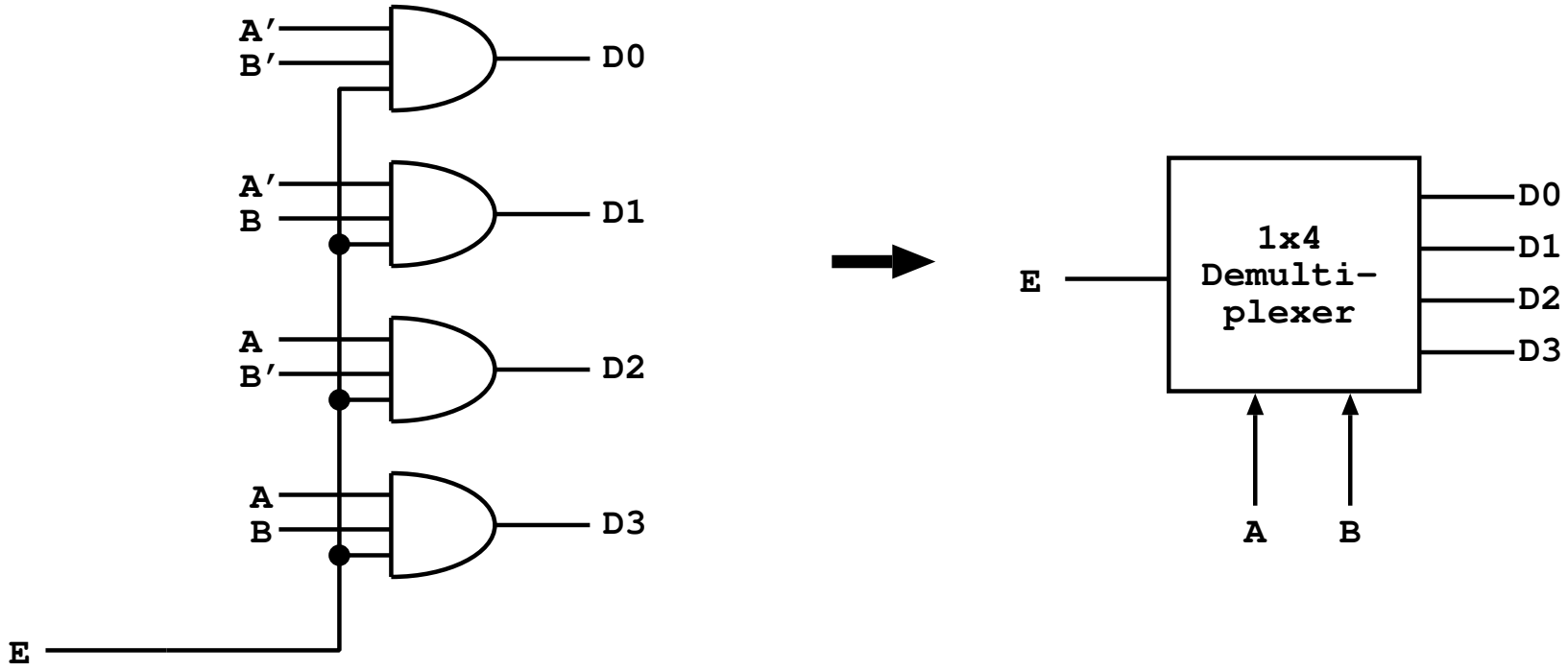
- enable 이 첨가된 decoder



  - 진리표 및 논리도 (* 그림 3-13)

디코우더 (decoder, 복호기) (Cont'd)

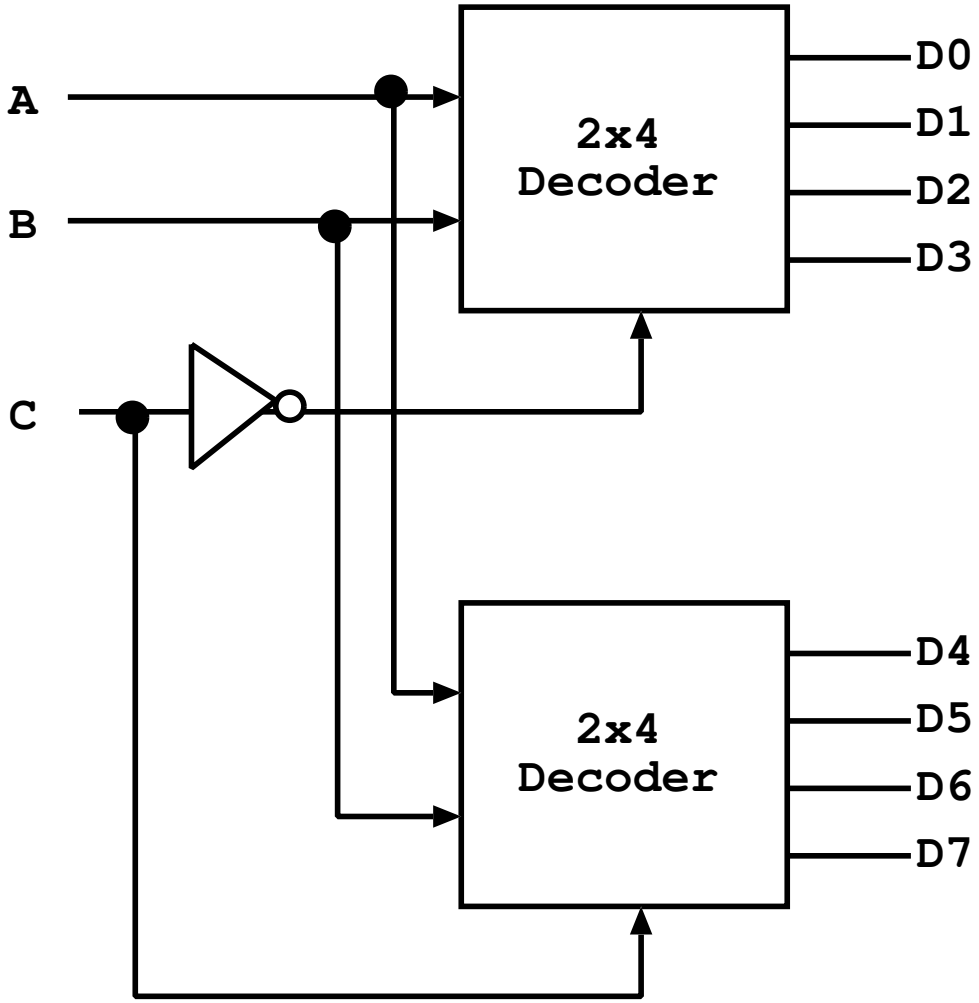
– enable 이 있는 2 × 4 decoder 는 1 × 4 demultiplexer 역할을 함



디코우더 (decoder, 복호기) (Cont'd)

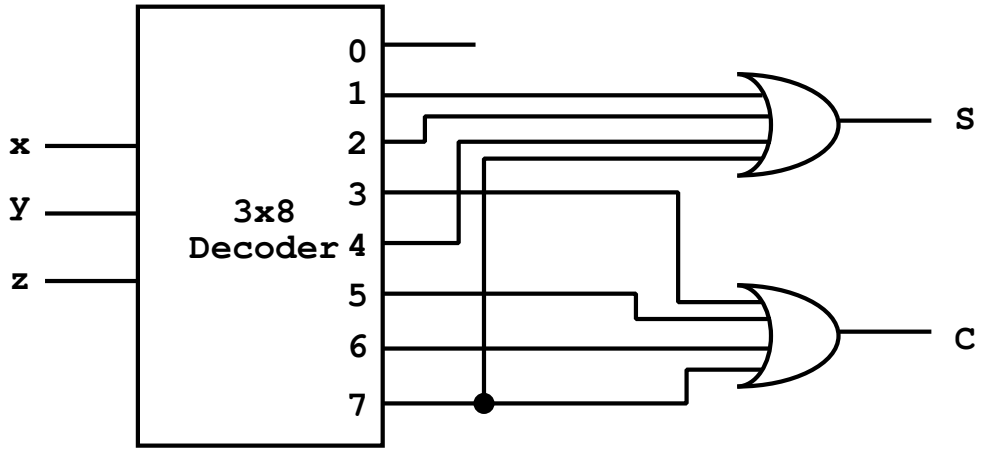
- enable 이 있는 2×4 decoder 2개를 이용 3×8 decoder 의 설계 (*그림 3-14)

디코우더 (decoder, 복호기) (Cont'd)



디코우더에 의한 조합논리 구현

- decoder 의 출력은 민텀항을 표시하기 때문에 임의의 조합논리를 구현할 수 있음
- 출력이 1 인 민텀항을 OR 로 묶음으로 해결
- 예) 전가산기를 decoder 와 OR gate 로 구현하라
  - 전가산기
    - S(x,y,z) = $\sum$ (1,2,4,7)
    - C(x,y,z) = $\sum$ (3,5,6,7)
  - 설계그림 (*그림 3-15)



인코우더 (encoder, 부호기)

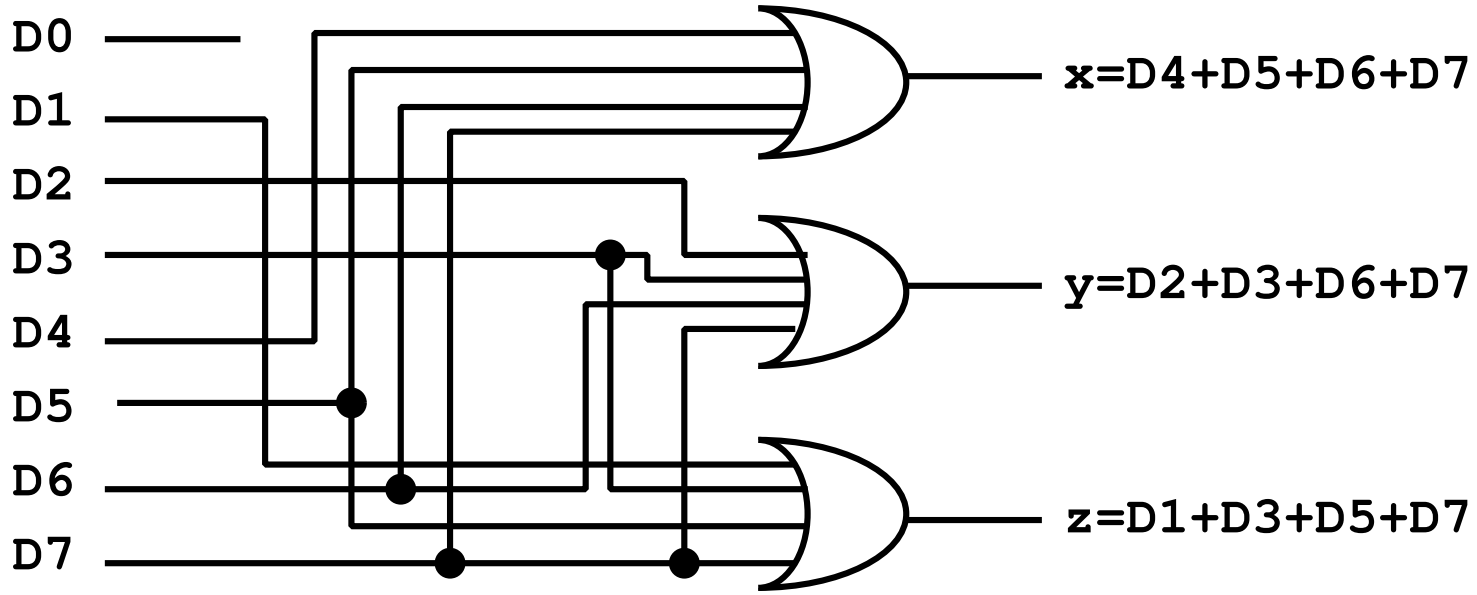
- encoder 는 정보를 code 화하는 것이다.
- 정보는 여러개의 선으로 들어오며 각각 0과 1의 값을 갖는다
- 단 동시에 두 개의 선이 1이 되는 경우는 없다고 가정한다.
- 2^n 개의 입력 선에 대하여 n bit 의 code 가 출력된다
- 하나 이상의 입력이 1 이 되는 것은 모두 리던던시이다.
- 진리표 : *표 3-5

인코우더 (encoder, 부호기) (Cont'd)

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

인코우더 (encoder, 부호기) (Cont'd)

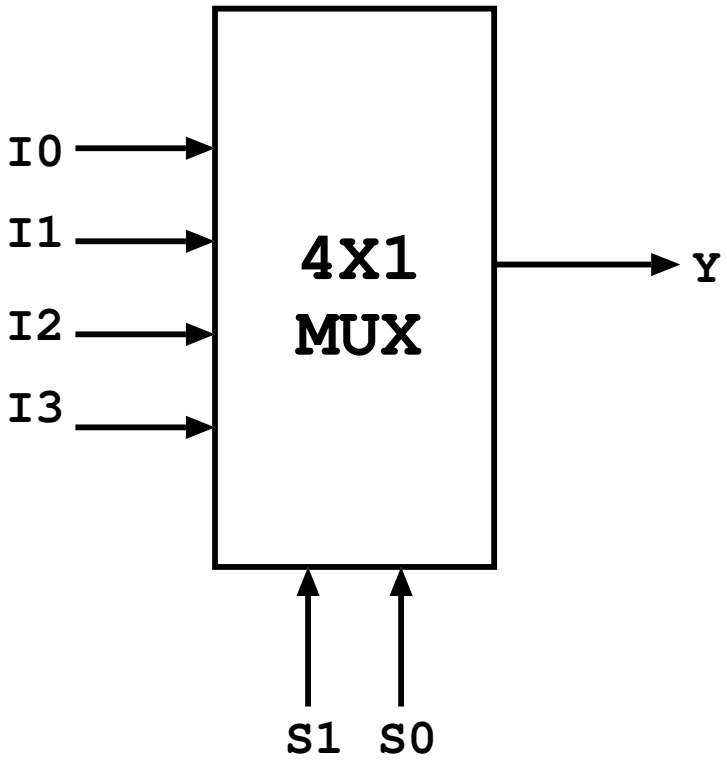
- 논리도 구현



- 우선 순위 인코더
  - 두 개 이상의 입력이 1이 되었을 때 높은 우선 순위의 것만을 코드화함
  - 진리표 (*표 3-6)
  - 논리도 (*그림 3-17)

Multiplexer (MUX)

- 병렬 데이터 →직렬데이터
- MUX (4×1) 의 설계

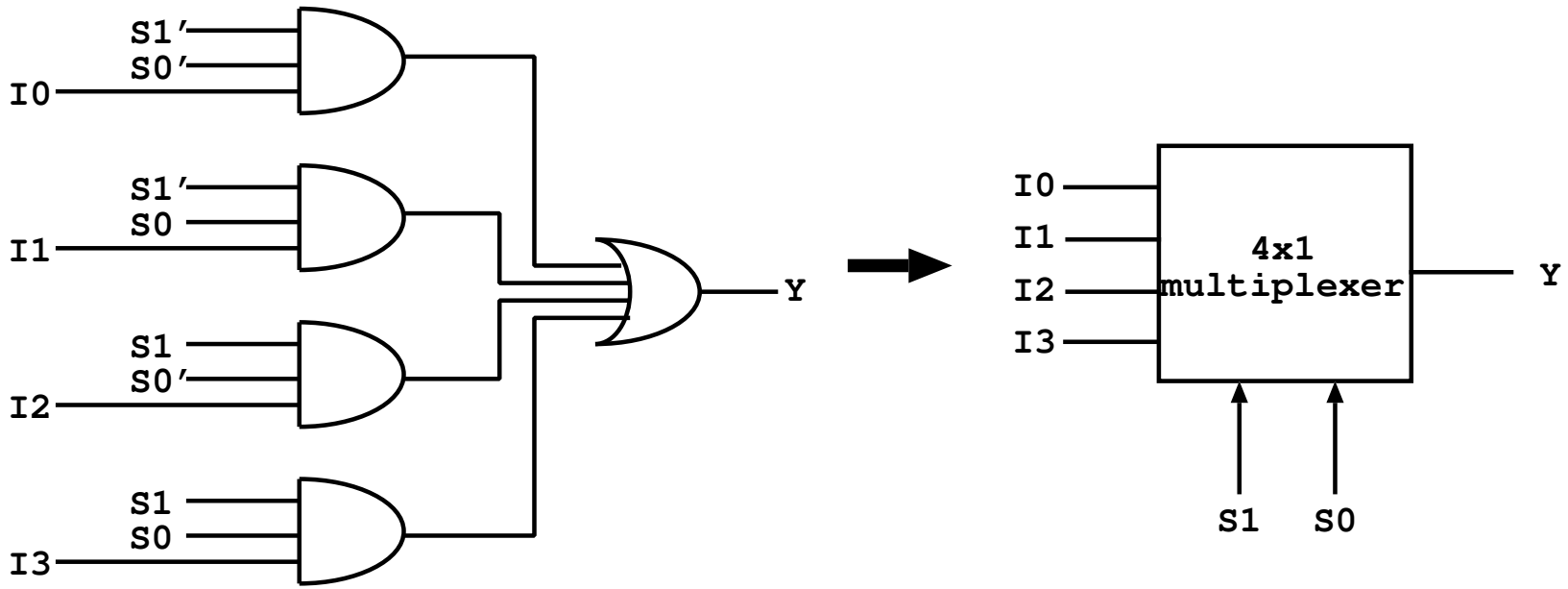


– 진리표

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

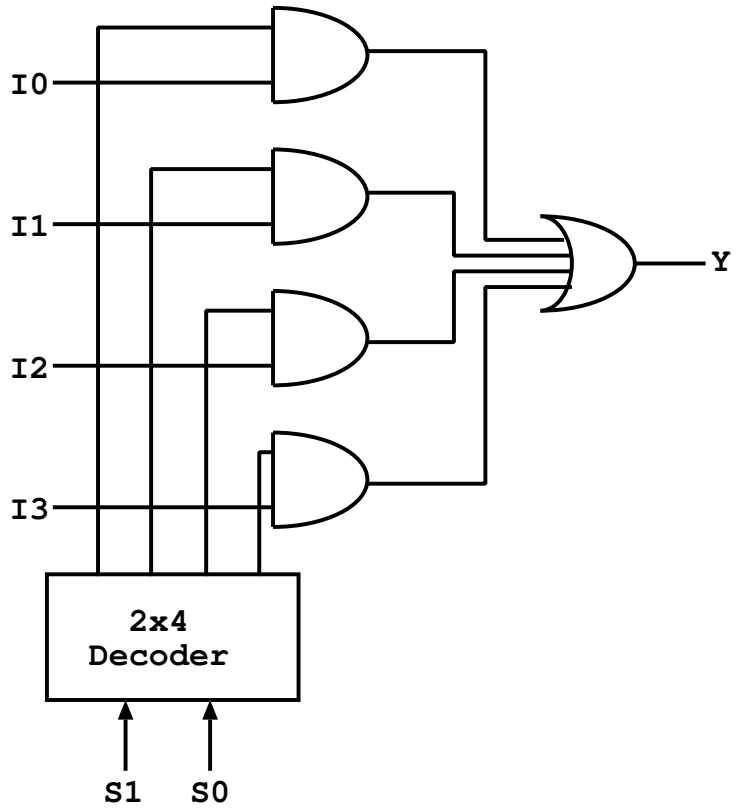
Multiplexer (MUX) (Cont'd)

– 논리도 (*그림 3-18)



Multiplexer (MUX) (Cont'd)

– Decoder 를 이용한 MUX 의 구현 (선택 선을 decoder 로 대치할 수 있다)



– 전송게이트를 갖는 4-to-1 line 멀티플렉서 (*그림 3-19)

– 4개의 2-to-1 line 멀티플렉서 (*그림 3-20)

MUX 의 응용

- MUX 에 의한 부울함수의 구현
- 예) 다음 진리표를 구현하라

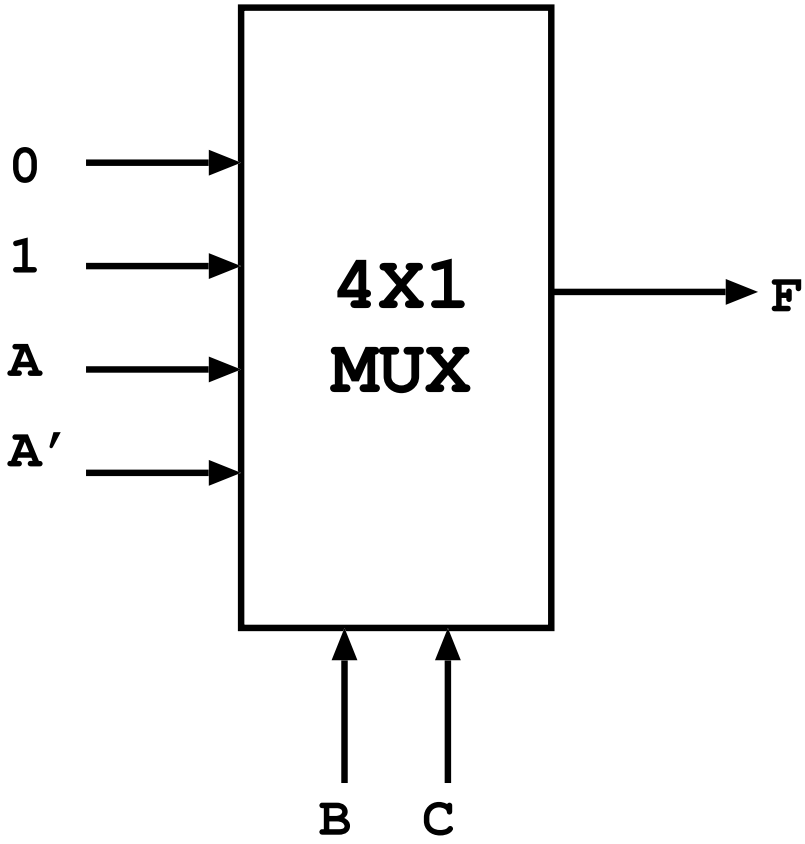
민텀	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

MUX 의 응용 (Cont'd)

– 실현 표를 만듦

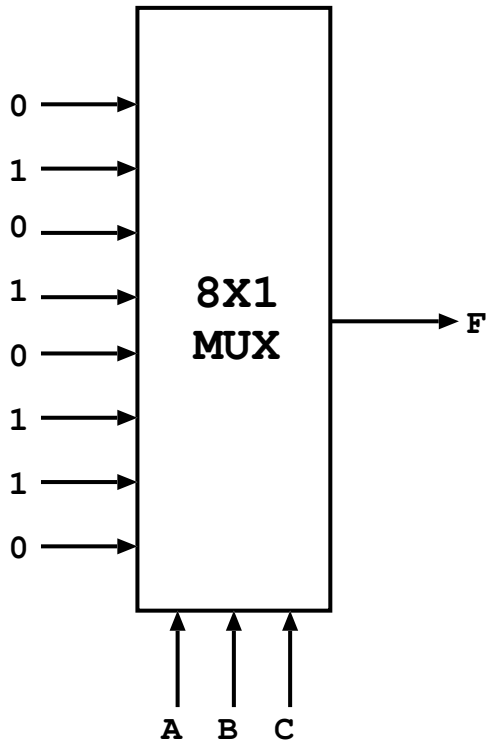
	I0	I1	I2	I3
A′	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	A′

– 4×1 MUX 로 구현



MUX 의 응용 (Cont'd)

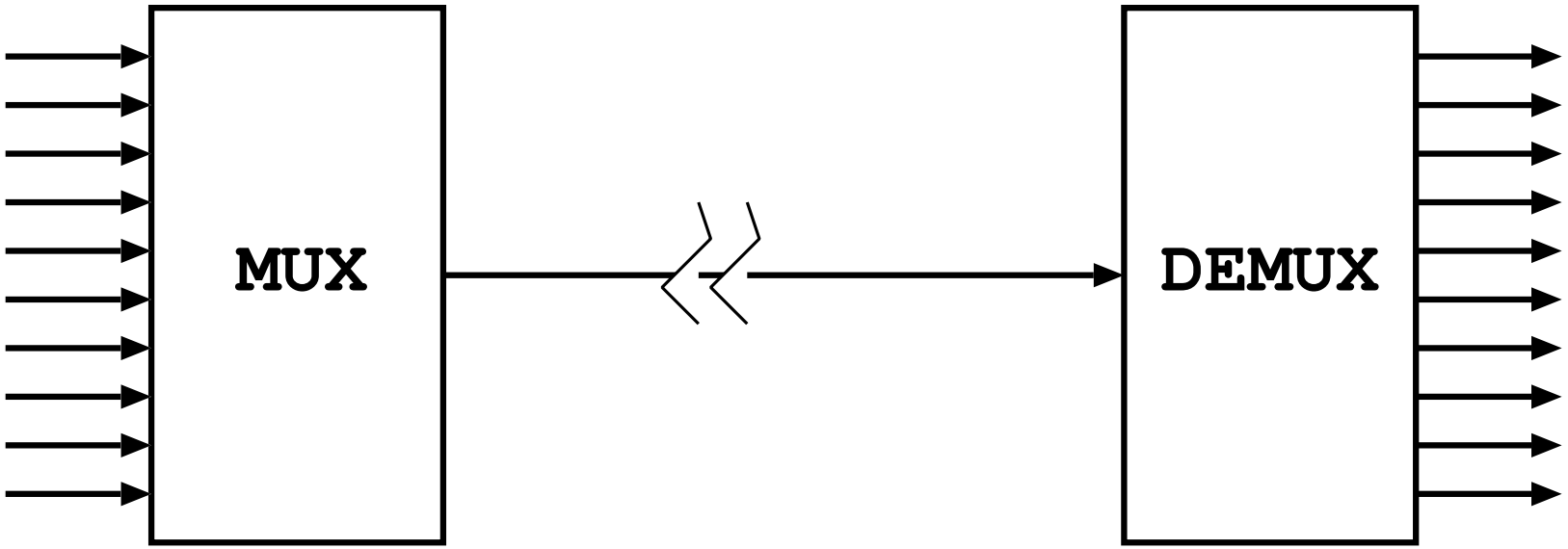
- 8×1 MUX 로 구현하면



- 예) *그림 3-21, 3-22

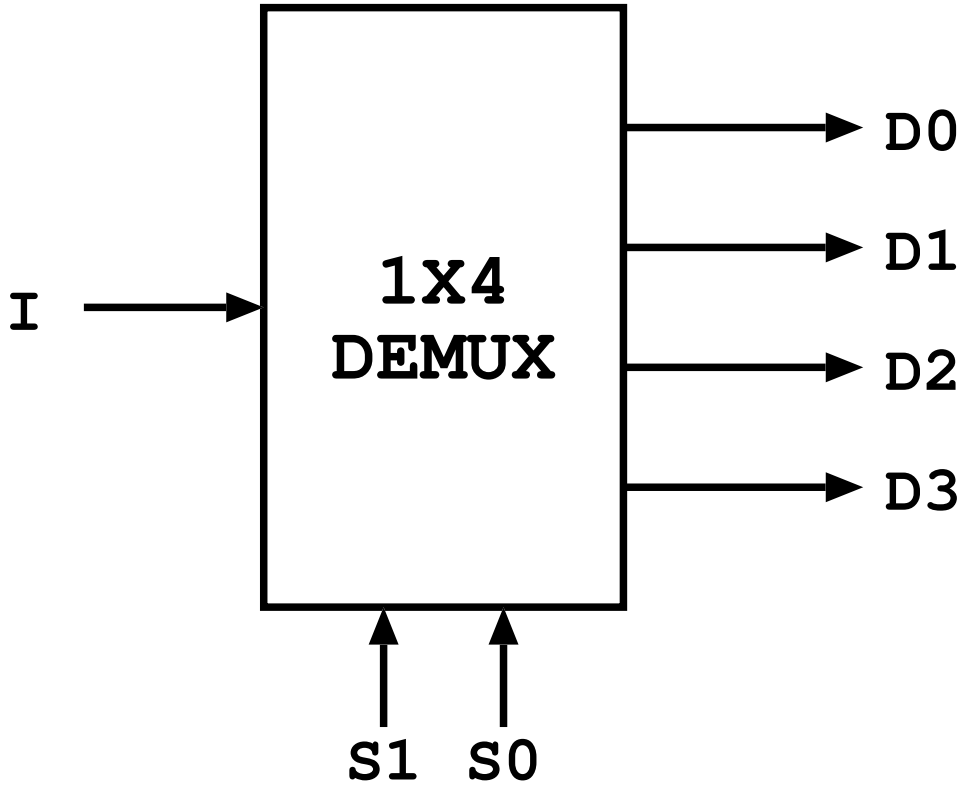
디멀티플렉서 (demultiplexer)

- demultiplexer (DEMUX) : 직렬데이터 →병렬데이터
- multiplexer (MUX) : 병렬데이터 →직렬데이터



디멀티플렉서 (demultiplexer) (Cont'd)

- DEMUX (1×4) 의 설계



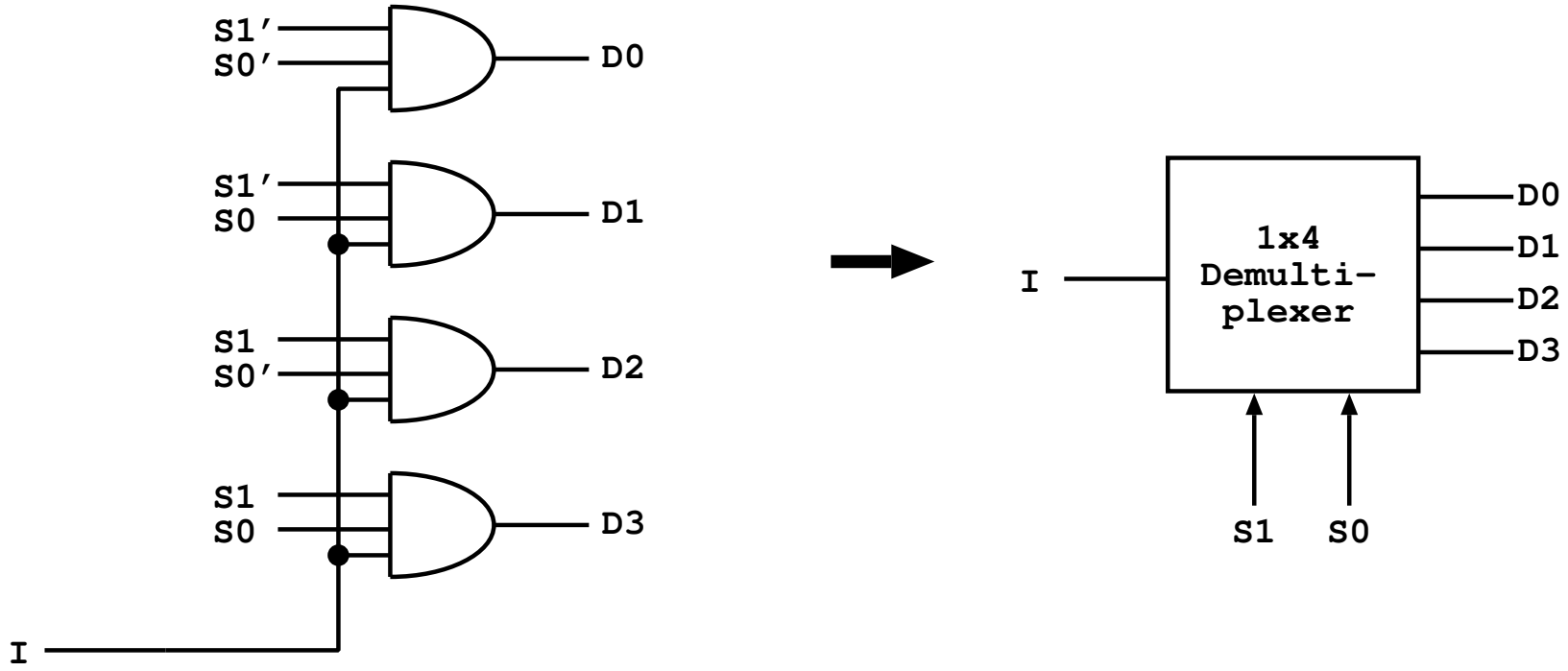
- 진리표

S_1	S_0	D_3	D_2	D_1	D_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

- 입력 정보를 특정 출력으로 보내기 위해 선택 선이 필요 : 2^n 입력에 대하여 n 개의 선택선

디멀티플렉서 (demultiplexer) (Cont'd)

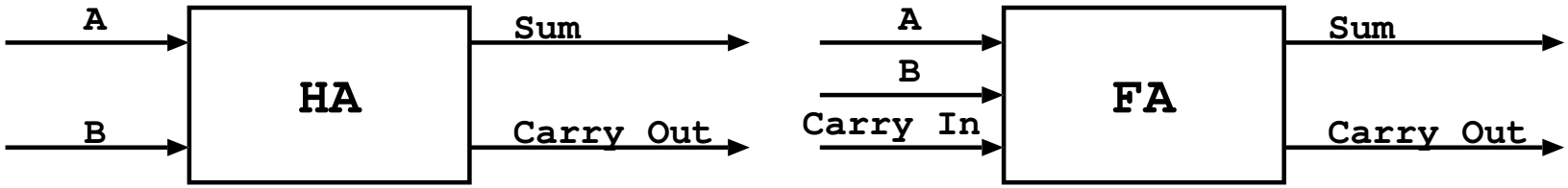
– 논리도



– 1-to-4 line 디멀티플렉서 (*그림 3-23)

2진 가산기

- 2bit 및 3bit 가산기를 설계함
- 두 종류의 가산기
  - 반가산기 (Half Adder : HA) : 2 bit 가산기
  - 전가산기 (Full Adder : FA) : 3 bit 가산기 →2 bit + 1 bit carry



- 예) 4bit 의 2진수 X, Y 를 더할때

```
C3 C2 C1 C0
   x3 x2 x1 x0
+  y3 y2 y1 y0
--------------
   S3 S2 S1 S0
   FA FA FA HA
```

2진 가산기 (Cont'd)

- 반가산기의 설계
  - 진리표를 작성

입력		출력	
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

  - 간략화된 부울함수를 얻음

$C = xy$

$S = x'y + xy' = (x'+y')(x+y) = (x'y'+xy)' = x \oplus y$

2진 가산기 (Cont'd)

– 논리도를 그림

S

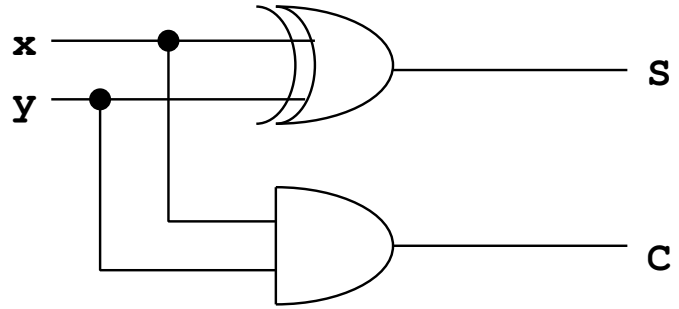
x \ y	0	1
0		1
1	1	

S = x'y+xy'
= x⊕ y

C

x \ y	0	1
0		
1		1

C = xy



2진 가산기 (Cont'd)

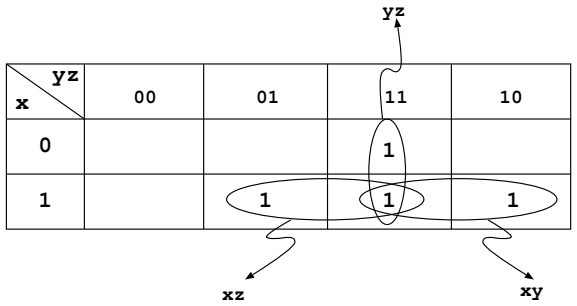
- 전가산기의 설계
  - 진리표를 작성

입력			출력	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

  - 논리도를 그림
  - 간략화된 부울함수를 얻음
    - S 에 대한 카르노 맵

x \ yz	00	01	11	10
0		1		1
1	1		1	

    - 결국 : $S = x'y'z + x'yz' + xy'z' + xyz = z \oplus (x \oplus y)$
    - C 에 대한 카르노 맵



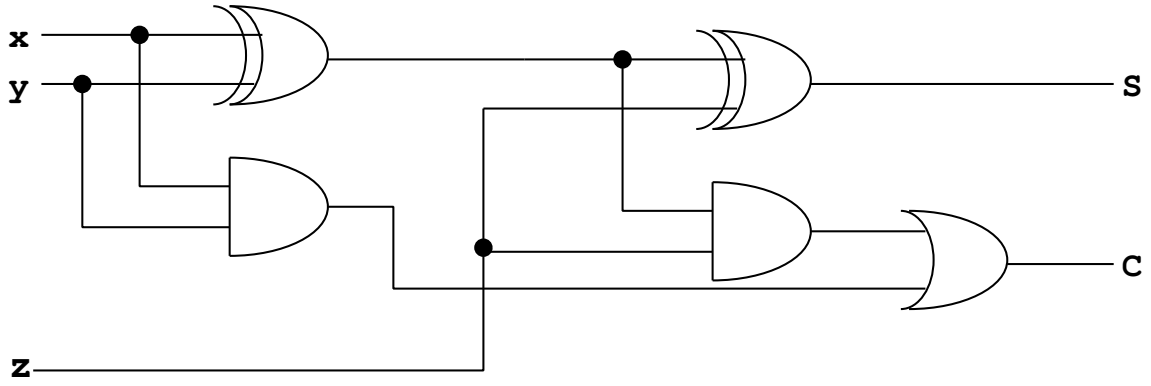
    - 결국 : $C = xy + xz + yz = xy'z + x'yz + xy = z(xy' + x'y) + xy$

2진 가산기 (Cont'd)

* 반가산기를 이용한 전가산기의 구현

$$S = x'y'z + x'yz' + xy'z' + xyz$$
$$= z \oplus (x \oplus y)$$

$$C = xy + xz + yz$$
$$= xy'z + x'yz + xy$$
$$= z(xy' + x'y) + xy$$
$$= z(x \oplus y) + xy$$



* 논리도 (*그림 3-26)

☛ 반감산기와 전감산기 위와 같이 설계할 수 있다.

- 2진 리플캐리 가산기 (*그림 3-27)

2진 가산기 (Cont'd)

- 캐리 예견 가산기
  - 위와 같은 리플캐리 가산기에서 연산시간 : C5 carry 연산시간으로 한정됨
  - 각 비트에서 carry를 미리 계산 (Look-ahead carry)
    - (carry propagate) Pi = Ai $\oplus$Bi
    - (carry generator) Gi = AiBi 라고 정하면
    - 전가산기는
      - Si = Ai $\oplus$Bi $\oplus$Ci = Pi $\oplus$Ci
      - Ci+1 = Ci(Ai $\oplus$Bi) + AiBi = CiPi + Gi 가됨
    - 그러므로 C1 이 주어지면
      - C2 = C1P1 + G1
      - C3 = C2P2 + G2 = (C1P1 + G1)P2 + G2 = C1P1P2 + G1P2 + G2
      - C4 = C3P3 + G3 = C1P1P2P3 + G1P2P3 + G2P3 + G3
        즉 상위캐리를 C1과 두비트로 바로 만들 수 있다.
    - Look-ahead carry generator 를 이용한 4비트 전가산기 (*그림 3-28)

2진 가산기-감산기

- 부호를 갖는 2진수 (*표 3-9)
  - 부호화된 절대값 표현 (signed-magnitude) : 부호와 크기로 표현
  - 부호화된 보수 표현 (signed-complement) : 부호와 보수(음수일경우)로 표현
  - 가산-감산기 회로 (*그림 3-30)
- 부호화된 2진 가산과 감산 (예제 3-8)
- 오버플로우
  - 2개의 n 비트의 합이 $n+1$ 비트를 차지할때 발생
  - 부호화 하였을경우는 최상위비트로 캐리 와 최상위비트에서 캐리중 1의 갯수가 홀수일때 발생
  - 오버플로우 검출회로 (그림 3-31)
- 2진 곱셈기
  - 2진 곱셈기의 논리도 (*그림 3-32)
  - 4bit $\times$ 3bit 2진 곱셈기 (*그림 3-33)

10진 산술연산

- 두 개의 십진수를 더하는 가산기: 9개의 입력 5개의 출력이 필요
- 논리설계방법으로는 너무 복잡
- 두 개의 4bit 2진가산기를 이용
  - 두 개의 한자리 십진수 A,B 를 더함

```
  -십진수-          ---이진수---
     Ci                      Ci
     A             A4 A3 A2 A1
   + B           + B4 B3 B2 B1
  -----           -----------
 Ci+1 S       Ci+1 S4 S3 S2 S1
```

  - 두 개의 십진수를 더했을 때 최대크기 : $9 + 9 + 1$ (carry) $= 19$

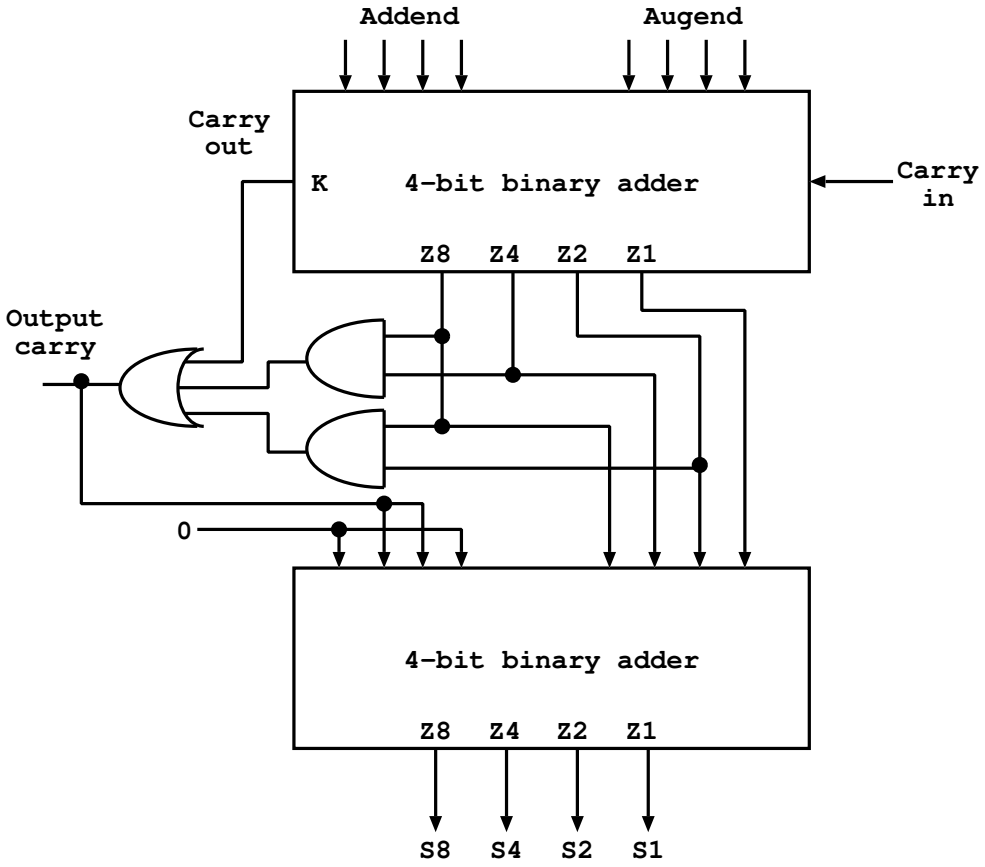
10진 산술연산 (Cont'd)

- 설계방침
  - * 두 개의 십진수를 더한 결과가 9보다 작으면 그대로 출력, carry 없음
  - * 두 개의 십진수를 더한 결과가 9보다 크면 10진수로 바꿈 : 6 (0110) 을 더함, carry 있음
  - * 결국 carry 가 있으면 0110 을 더함, 없으면 그대로 출력
  - * 진리표에서 carry 가 발생하는 조건을 구함
    - · Ci+1 = K + Z8Z4 + Z8Z2
  - 구현 그림 (*그림 3-34)

- 진리표

2진수로 표현된 합					BCD 합					10 진수
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

10진 산술연산 (Cont'd)



표준 그래픽기호

- 4비트 병렬가산기에 대한 표준 그래픽기호 (*그림 3-35)
- 디코더에대한 표준 그래픽기호 (*그림 3-36)
- 멀티플렉서에서 G-AND 종속 예 (*그림 3-37)
- 멀티플렉서에 대한 표준 그래픽 기호 (*그림 3-38)
- 가산기-감산기에 대한 그래픽 기호 (*그림 3-39)
- 입력과 출력에 연관된 기호 제한 (*그림 3-40)
- Homework
  - 3장 연습문제
    - 1, 3, 4, 6, 10, 14, 16, 17, 22, 26, 27, 41