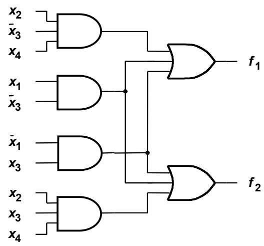
Multiple-Output Functions

x_3x_4 \ x_1x_2	00	01	11	10
00			1	1
01			1	1
11	1	1	1	
10	1	1		

$f_1 = x_1 x_3' + x_1' x_3 + x_2 x_3' x_4$



x_3x_4 \ x_1x_2	00	01	11	10
00			1	1
01			1	1
11	1	1	1	
10	1	1		

$f_2 = x_1 x_3' + x_1' x_3 + x_2 x_3 x_4$

Multiple-Output Functions

- **f = a'b'c'd + a'b'cd + a'bc'd + a'bcd + abcd + ab'c'd'**
- **g= abcd + ab'c'd' + ab'cd + ab'cd'**

f

ab \ cd	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	0	1	0
10	1	0	0	0

g

ab \ cd	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	1	0	1	1

- **Want to implement the two functions at the same time**

First Attempt: Minimize Functions Separately

f

ab \ cd	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	0	1	0
10	1	0	0	0

g

ab \ cd	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	1	0	1	1

- **f = a'd + bcd + ab'c'd'** — **Cost = 16 (3 AND, 1 OR, 12 inputs)**
- **g = acd + ab'd'** — **Cost = 11 (2 AND, 1 OR, 8 inputs)**

Nothing can be shared!!! **Total cost = 27**

Better Approach: See what can be shared

f

ab \ cd	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	0	1	0
10	1	0	0	0

g

ab \ cd	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	1	0	1	1

- **f = a'd + abcd + ab'c'd'**
- **g = abcd + ab'c'd' + ab'c**

Only 4 distinct min terms

- **Total cost = 25 (4 AND, 2 OR, 19 inputs)**

Multilevel Synthesis

- **SOP or POS circuits consist of two levels (stages) of gates**
- **It may be possible to reduce fan-in or cost if more than 2 levels are used**
- **But delay may be increased**

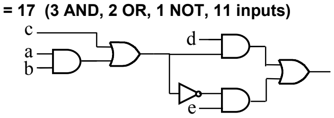
- **Consider h = abe + ce + de + abfg + cfg + dfg**
- **It is already in the simpliest SOP form**
- **Cost = 30 (6 AND, 1 OR, 23 inputs)**

- **By factoring,**
  h = (ab + c + d).e + (ab + c + d).fg
  = (ab + c + d).(e + fg)
- **Cost = 16 (3 AND, 2 OR, 11 inputs)**

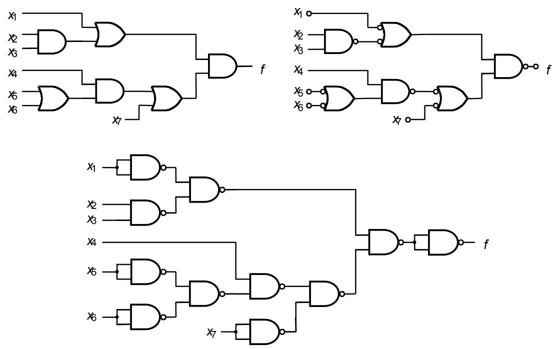
Another Example of Multilevel Synthesis

- **Consider f = abd + cd + a'c'e + b'c'e**
- **Cost = 20 (4 AND, 1 OR, 15 inputs)**

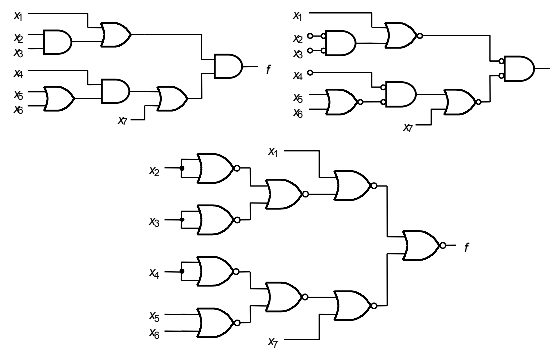
- **By Boolean algebra,**
  f = (ab + c).d + (a'+b')c'e
  = (ab + c).d + (ab)'c'e
  = (ab + c).d + (ab + c)'.e
- **Note that the sub-function ab + c appears twice in f**
- **We can make the implementation of f simpler by building a sub-circuit for ab + c.**
- **Cost = 17 (3 AND, 2 OR, 1 NOT, 11 inputs)**



Multilevel NAND Circuit

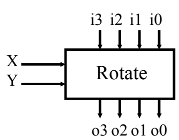


Multilevel NOR Circuit



Application of Multiplexer: 4-bit rotator circuit

- **Inputs are i3 i2 i1 i0; Outputs are o3 o2 o1 o0**
- **Output is the input rotated by 0, 1, 2 or 3 bit positions**
- **Rotation is to the left**
- **X Y determines # of positions rotated to the left:**
  - **If X Y = 0 0 (left rotate 0 bit position), o3 o2 o1 o0 = i3 i2 i1 i0**
  - **If X Y = 0 1 (left rotate 1 bit position), o3 o2 o1 o0 = i2 i1 i0 i3**
  - **If X Y = 1 0 (left rotate 2 bit positions), o3 o2 o1 o0 = i1 i0 i3 i2**
  - **If X Y = 1 1 (left rotate 3 bit positions), o3 o2 o1 o0 = i0 i3 i2 i1**

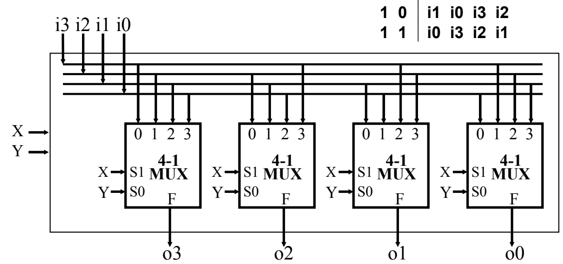


X Y	Output o3 o2 o1 o0
0 0	i3 i2 i1 i0
0 1	i2 i1 i0 i3
1 0	i1 i0 i3 i2
1 1	i0 i3 i2 i1

Implementation of Rotator by Multiplexers

- **We can implement a 4-bit rotator using four 4-to-1 multiplexers**

X Y	Output o3 o2 o1 o0
0 0	i3 i2 i1 i0
0 1	i2 i1 i0 i3
1 0	i1 i0 i3 i2
1 1	i0 i3 i2 i1



Encoder

- **Encoders perform the inverse function of Decoders.**
- **An encoder has 2^n (or less) input bits and n output bits**
- **The output bits generate the binary code corresponding to the input value**
- **Assuming only one input has a value of 1 at any given time**
- **Example: An 8-to-3 Encoder**

Inputs								Outputs		
D7	D6	D5	D4	D3	D2	D1	D0	A2	A1	A0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

A2=D4+D5+D6+D7
A1=D2+D3+D6+D7
A0=D1+D3+D5+D7

Priority Encoders

- **Each input signal has a priority level associated with it**
- **May have more than one 1's in the input signals**
- **Outputs indicate the active input that has the highest priority**
- **Example: 4-to-2 priority encoder**
  - **Assume w3 has the highest priority and w0 the lowest**
  - **y1 y0 indicate the active input with highest priority**
  - **z indicates none of the inputs is equal to 1**

w3	w2	w1	w0	y1	y0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

x: both 0 and 1 (irrelevant)

Let i0 = w0 w1' w2' w3'
i1 = w1 w2' w3'
i2 = w2 w3'
i3 = w3
Then y0 = i1 + i3
y1 = i2 + i3