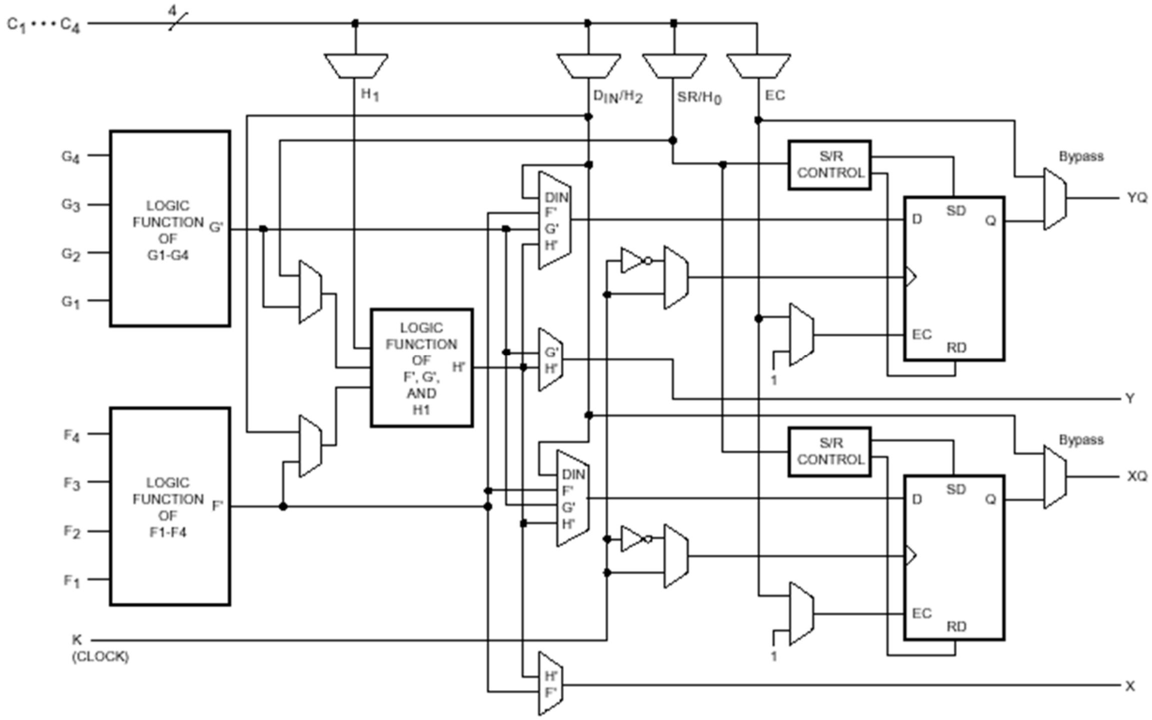
*ECE 428   Programmable ASIC Design*

# FPGA Implementation of Combinational Logic

Haibo Wang
ECE Department
Southern Illinois University
Carbondale, IL 62901

# Combinational Logic Implemented by Xilinx XC4000 CLB



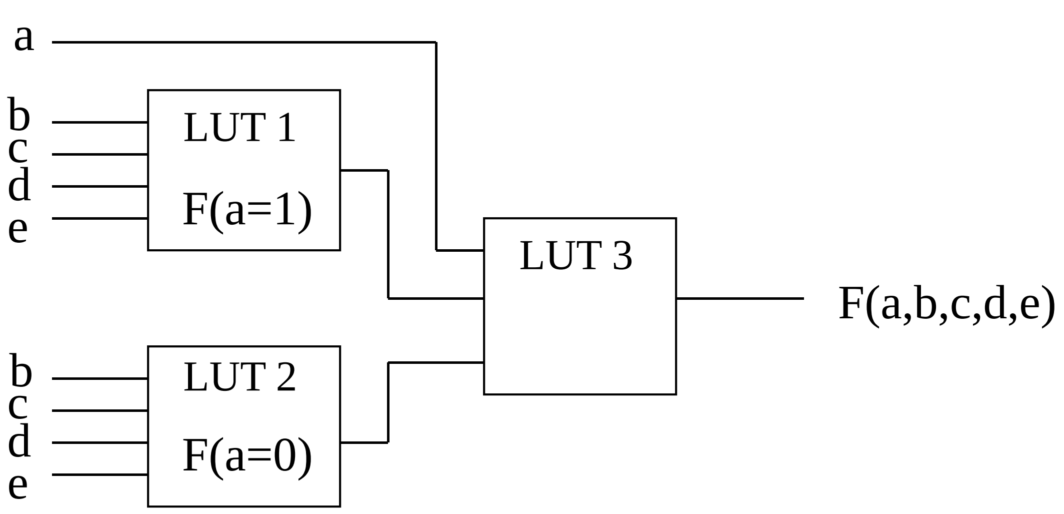

- Any function of up to four variables, plus any second function of up to four unrelated variables, plus any third function of up to three unrelated variables
- Any single function of five variables
- Any function of four variables together with some functions of six variables
- Some functions of up to nine variables.

# Combinational Logic Implemented by Xilinx XC4000 CLB

- Why can any five-input function be implemented by a XC4000 CLB?

$$F(a, b, c, d, e) = a \bullet F(a=1) + a' \bullet F(a=0)$$

— Both F(a=1) and F(a=0) are four-input functions



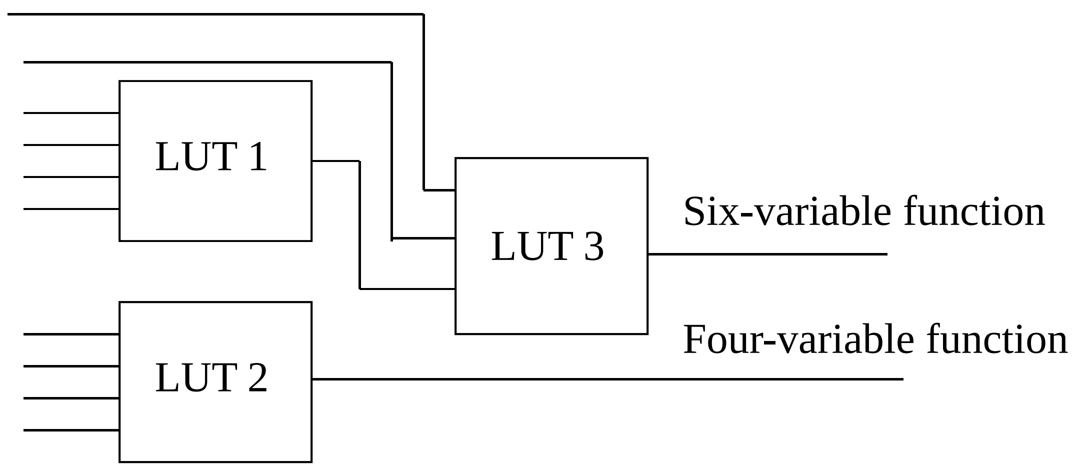
# Combinational Logic Implemented by Xilinx XC4000 CLB

- Any function of four variables together with some functions of six variables can be implemented by a single CLB
  - What kind of six variable functions can be implemented with another four variable function within a single CLB?

$$F(a,b,c,d,e,f) = a \cdot b \cdot F1 + a \cdot b' \cdot F2 + a' \cdot b \cdot F3 + a' \cdot b' \cdot F4$$

F1 = F(a=1, b=1); F2 = F(a=1, b=0);
F3 = F(a=0, b=1); F4 = F(a=0, b=0)

**Condition:** Among F1-F4, three of them are constant (e.g. F1=1, F2=F3=0)



# Example: A Six-Input Majority Function

- **Six-input majority function**: the function output is 1 if three or more than three inputs are 1.

$$F(a,b,c,d,e,f) = \sum_i m_i \quad (m_i \text{ Minterm of function F})$$

| Number of inputs with logic 1 | Minterms |
|---|---|
| 6 | a•b•c•d•e•f |
| 5 | a•b•c•d•e•f' a•b•c•d•e'•f a•b•c•d'•e•f a•b•c'•d•e•f<br>a•b'•c•d•e•f a'•b•c•d•e•f |
| 4 | a•b•c•d•e'•f' a•b•c•d'•e•f' a•b•c'•d•e•f' a•b'•c•d•e•f'<br>a'•b•c•d•e•f' a•b•c•d'•e'•f a•b•c'•d•e'•f a•b'•c•d•e'•f<br>a'•b•c•d•e'•f' a•b•c'•d'•e•f a•b'•c•d'•e•f a'•b•c•d'•e•f<br>a•b'•c'•d•e•f' a'•b•c'•d•e•f a'•b'•c•d•e•f |
| 3 | a•b•c•d'•e'•f' a•b•c'•d•e'•f' a•b'•c•d•e'•f' a'•b•c•d•e'•f'<br>a•b•c'•d'•e•f' a•b'•c•d'•e•f' a'•b•c•d'•e•f' a•b'•c'•d•e•f'<br>a'•b•c'•d•e•f' a'•b'•c•d•e•f' a•b•c'•d'•e'•f a•b'•c•d'•e'•f<br>a'•b•c•d'•e'•f a•b'•c'•d•e'•f' a'•b•c'•d•e'•f a'•b'•c•d•e'•f<br>a•b'•c'•d'•e•f a'•b•c'•d'•e•f a•b'•c•d'•e•f a'•b'•c'•d•e•f |

# Example: A Six-Input Majority Function

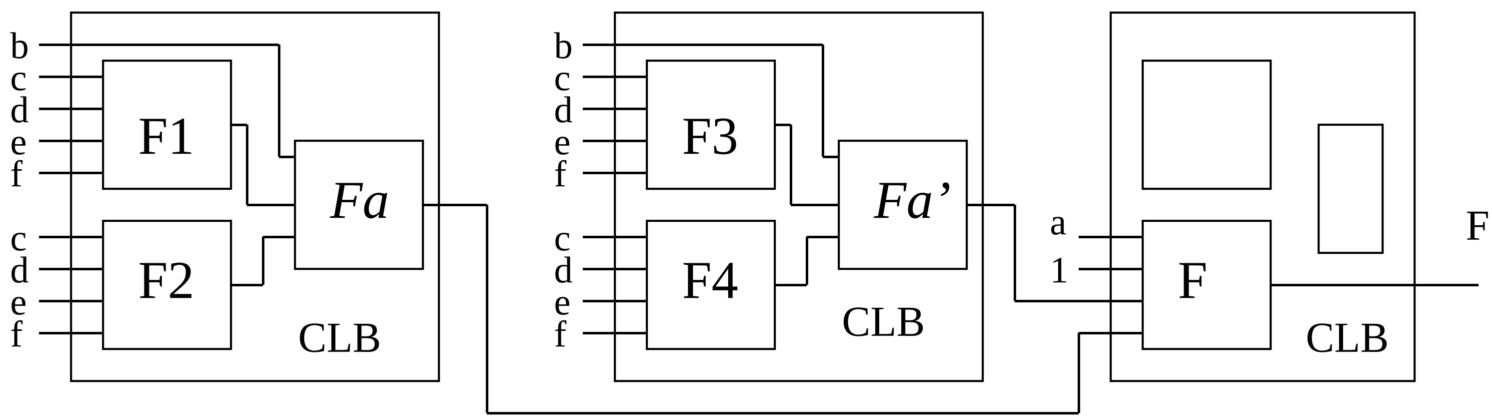
$F(a=1,b=1) = e+c+f \bullet c' + e' \bullet d$ (F1)

$F(a=1,b=0) = F(a=0,b=1) = c \bullet d + e \bullet f + d \bullet f + e \bullet d + f \bullet c + e \bullet c$ (F2, F3)

$F(a=0,b=0) = c \bullet d \bullet f + c \bullet d \bullet e + e \bullet f \bullet c + e \bullet f \bullet d$ (F4)

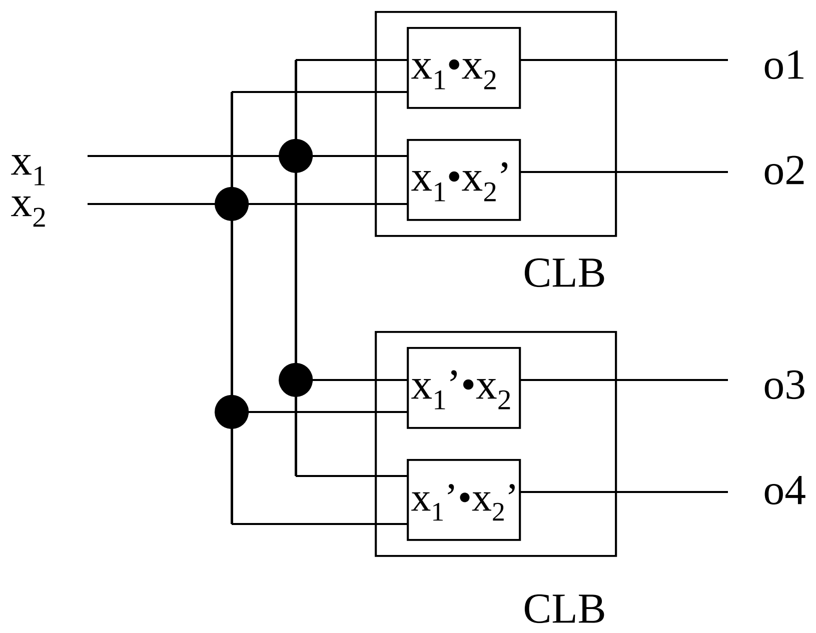
$$F = a \bullet b \bullet F(a=1,b=1) + a \bullet b' \bullet F(a=1,b=0) + a' \bullet b \bullet F(a=0,b=1) + a' \bullet b' \bullet F(a=0,b=0)$$

$$= a \bullet \underbrace{\{b \bullet F(a=1,b=1) + b' \bullet F(a=1,b=0)\}}_{Fa} + a' \bullet \underbrace{\{b \bullet F(a=0,b=1) + b' \bullet F(a=0,b=0)\}}_{Fa'}$$



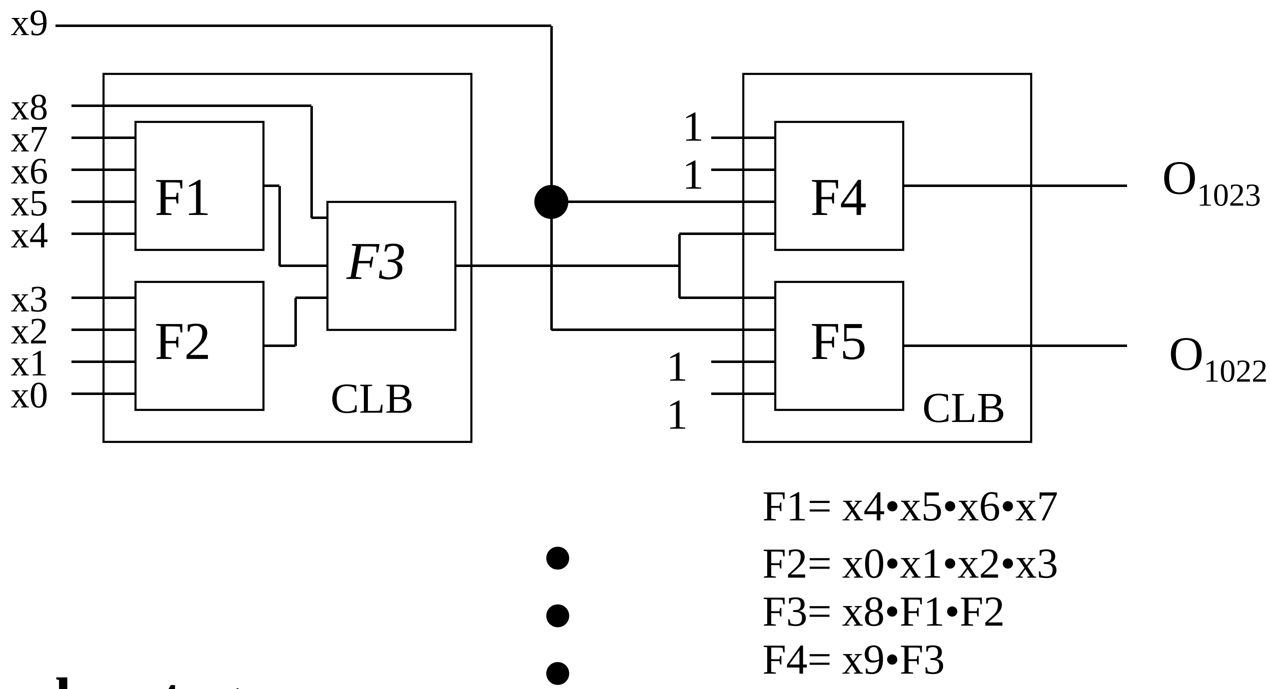
# Decoding Circuits

- 2-to-4 Decoding circuit



# Decoding Circuits

- ❑ 10-to-1024 Decoding circuit




F1= x4•x5•x6•x7

F2= x0•x1•x2•x3

F3= x8•F1•F2

F4= x9•F3

F5= x9'•F3

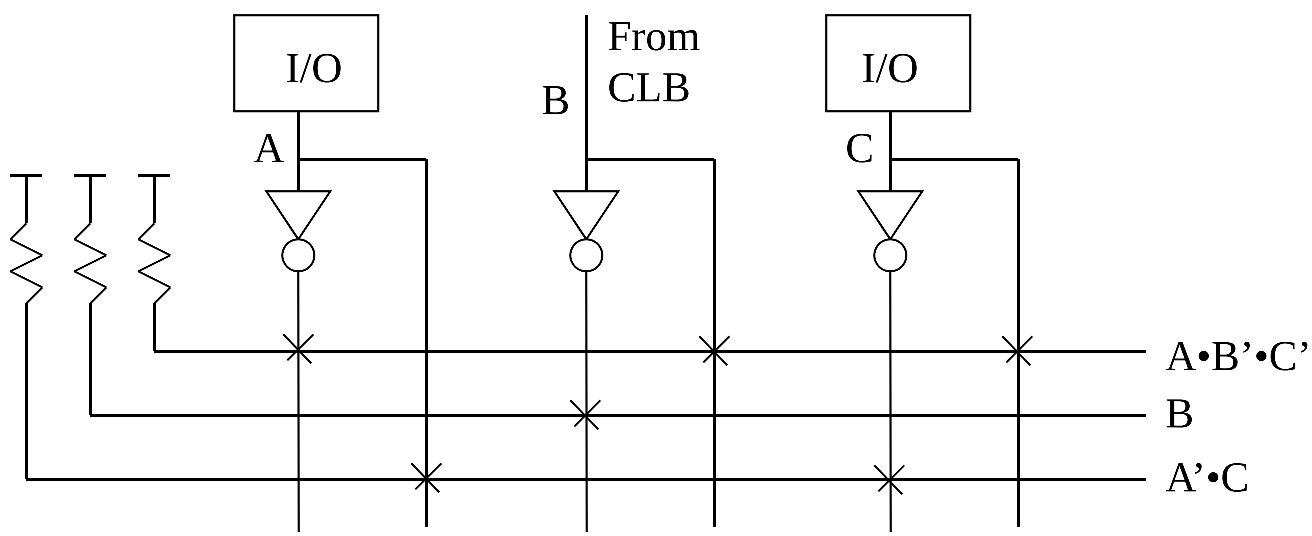
**Disadvantages**

— It needs 1024 CLBs; expensive to implement.

— It is a two level implementation, resulting large delay.

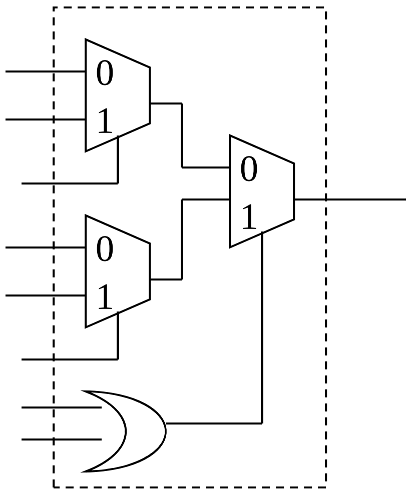
# Dedicated Decoding Circuits in Xilinx FPGAs

- Four dedicated programmable decoding circuits are included in Xilinx FPAGs.
- The number of decoder inputs ranges from 42 to 132 for different devices.
- The decoding circuits use wired-AND gate structures (like the AND plane in PAL).

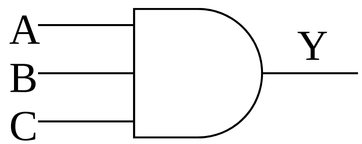


# Combinational Logic implemented by Actel ACT1

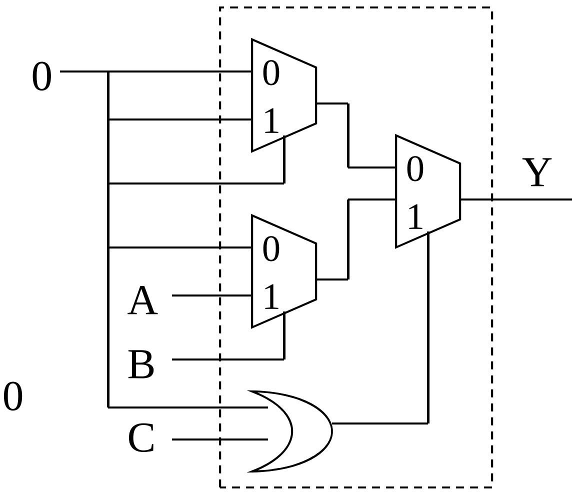
- Example




Actel ACT1 Cell




$$Y = A \bullet B \bullet C = C \bullet (AB) + C' \bullet 0$$
$$= C \bullet (B \bullet A + B' \bullet 0) + C' \bullet 0$$

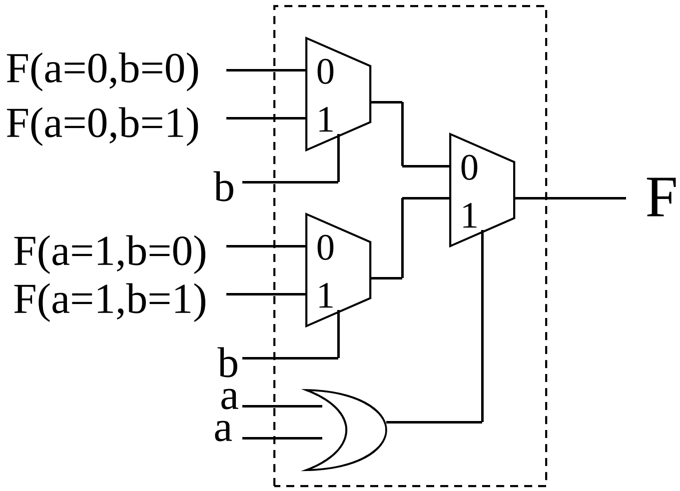



3-input AND gate implementation

# Combinational Logic implemented by Actel ACT1

- For a general three-input function

$$F(a, b, c) = a \bullet F(a=1) + a' \bullet F(a=0)$$
$$= a \bullet (b \bullet F(a=1,b=1) + b' \bullet F(a=1,b=0)) + a' \bullet (b \bullet F(a=0\ b=1) + b' \bullet F(a=0,b=0))$$




Note: F(a=1,b=1), F(a=1,b=0), F(a=0,b=1) and F(a=0,b=0) must be one of the the following values: c, c', 0, 1

- If for every variable its complement is also available, any three-input combinational function can be implemented by a single ACT cell.
- Some functions with inputs up to eight can be implemented by a single ACT1 cell.

# Example: A Six-Input Majority Function

- **Six-input majority function**: the function output is 1 if three or more than three inputs are 1.

  F(a=1,b=1,c=1) = 1 (F1)

  F(a=1,b=1,c=0) = F(a=1,b=0,c=1) =F(a=0,b=1,c=1) = e+f+d (F2,F3,F4)

  F(a=1,b=0,c=0) =F(a=0,b=1,c=0) = F(a=0,b=0,c=1) = e•f+d•f+e•d (F5,F6,F7)

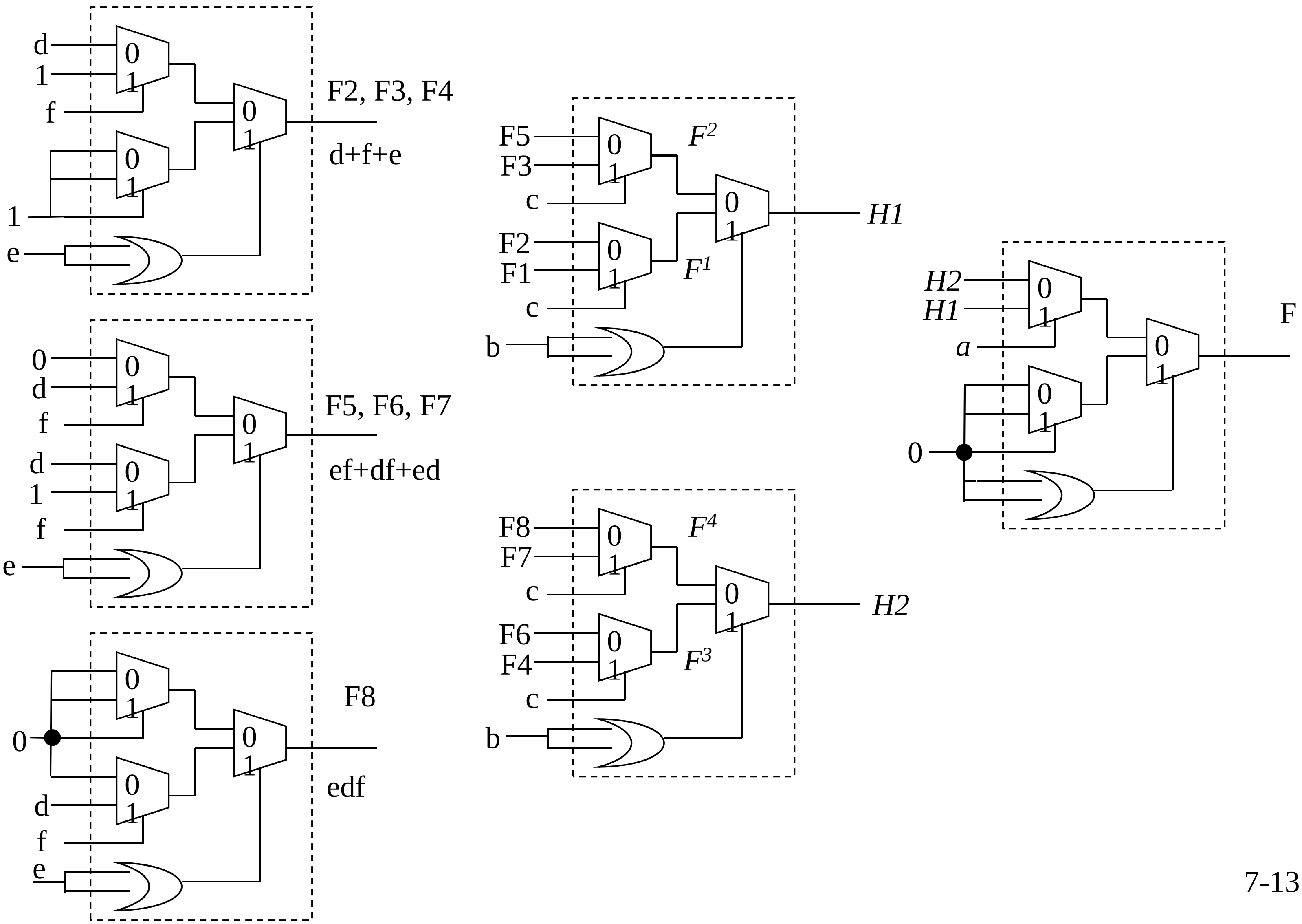
  F(a=0,b=0,c=0) = e•f•d (F8)

$$F = abcF1 + abc'F2 + ab'cF3 + a'bcF4 + ab'c'F5 + a'bc'F6 + a'b'cF7 + a'b'c'F8$$

$$= ab\underbrace{(cF1+c'F2)}_{F^1} + ab'\underbrace{(cF3+c'F5)}_{F^2} + a'b\underbrace{(cF4+c'F6)}_{F^3} + a'b'\underbrace{(cF7+c'F8)}_{F^4}$$

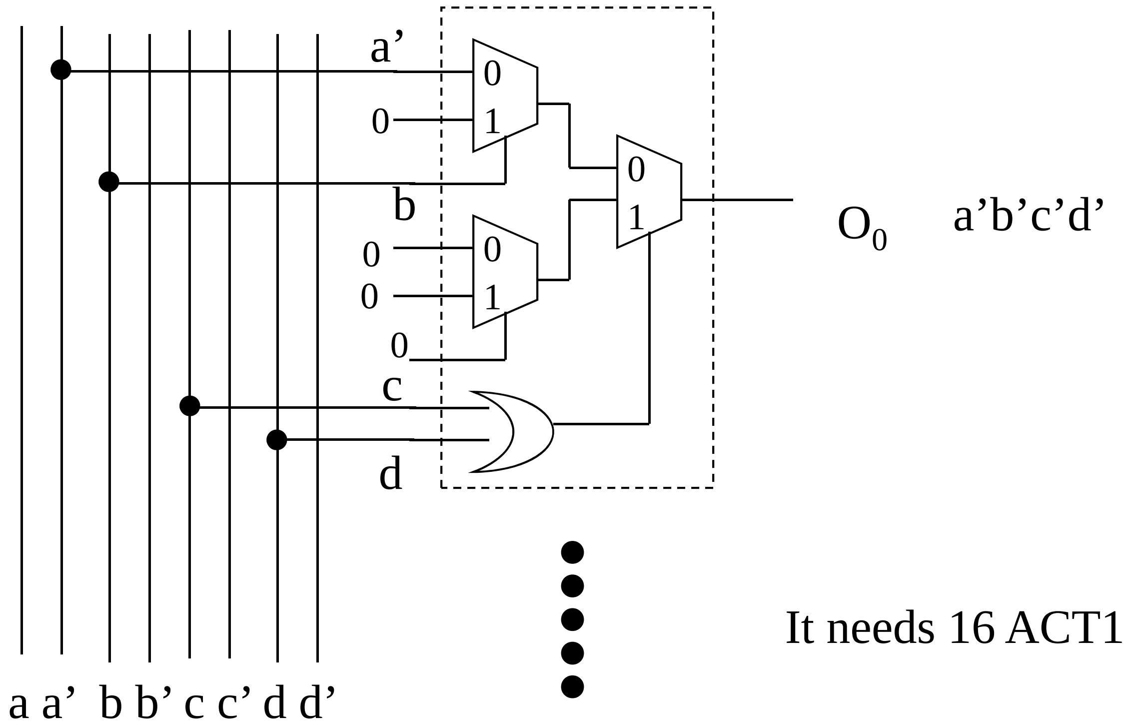
$$= a\underbrace{(bF^1 + b'F^2)}_{H1} + a'\underbrace{(bF^3 + b'F^4)}_{H2}$$

# Example: A Six-Input Majority Function



# Decoding Circuits

- ❑ 4-to-16 Decoding circuit

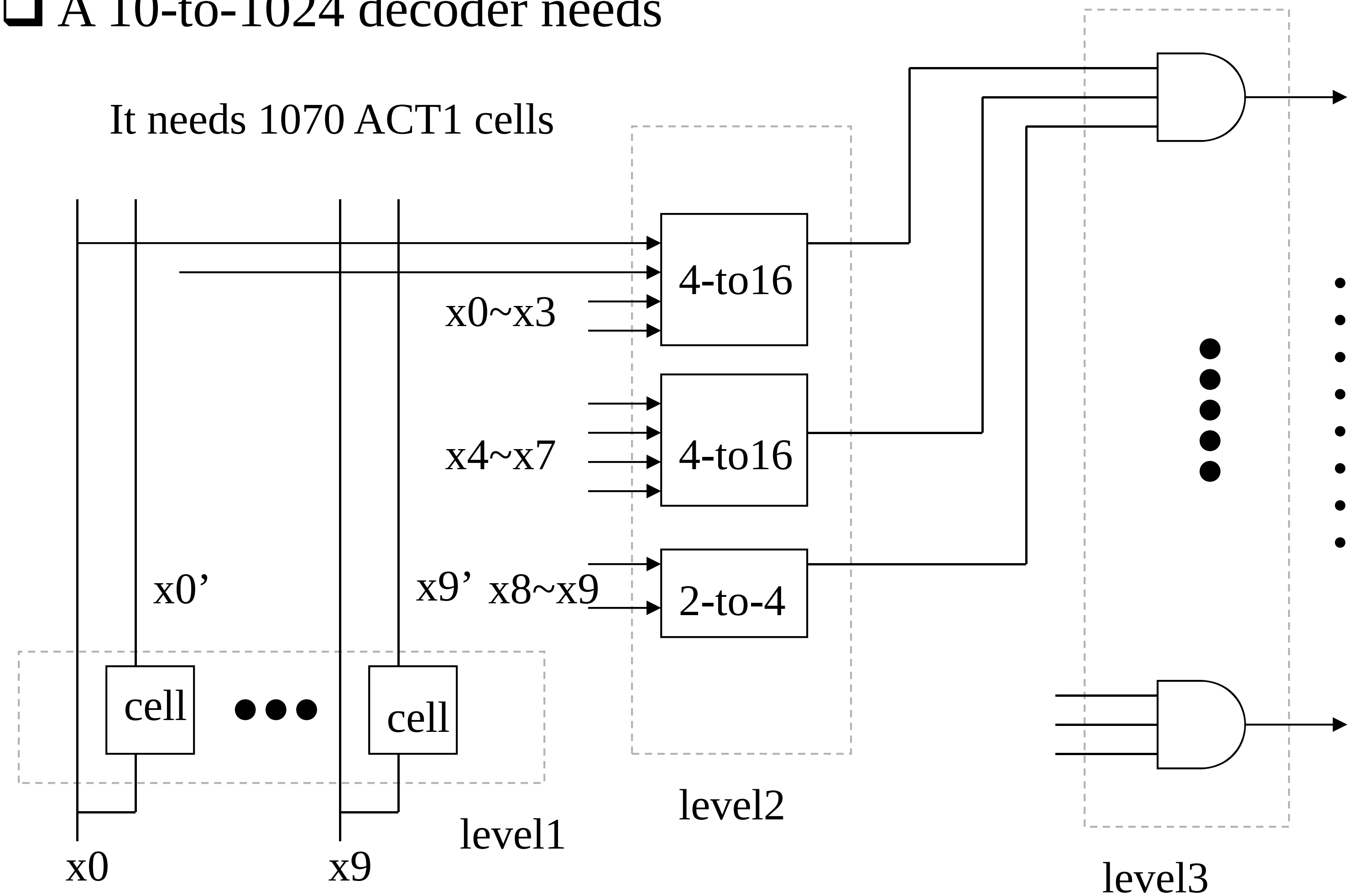



It needs 16 ACT1 cells

- ❑ Decoder circuits with more than four inputs need multi-level implementation.

# Decoding Circuits

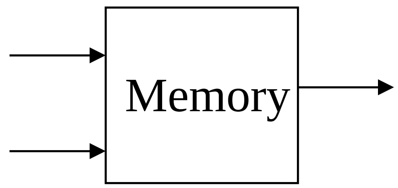
- A 10-to-1024 decoder needs



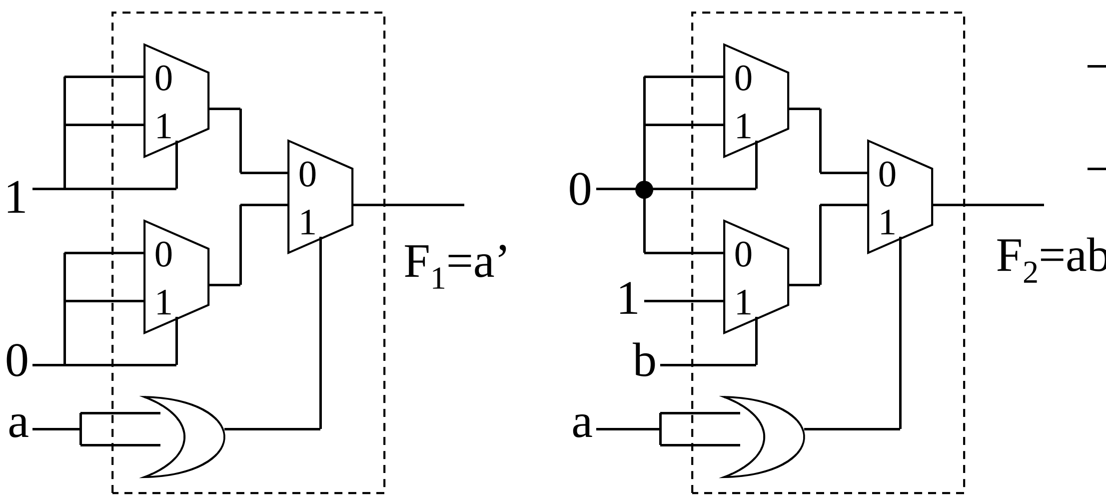

- Actel FPGAs also use dedicated decoding circuits.

# Speed Considerations

- For a combinational circuit implemented by a **single** memory block, its delay depends on neither the function type nor the input transition patterns.




- For a combinational circuit implemented by mux-based FPGAs, its delay normally depends on the function type and the input transition patterns.




— The delay of $F_1$ is smaller than that of $F_2$

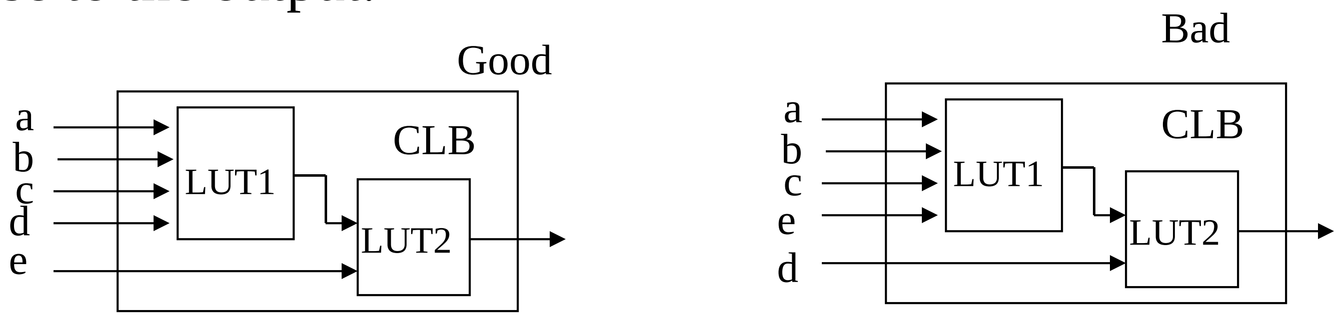
— For $F_2$, d1 is the output delay when inputs switch from a=1 and b=1 to a=0 and b=1; d2 is the delay when inputs switch from a=1 and b=1 to a=1 and b=0.

**d2 > d1**

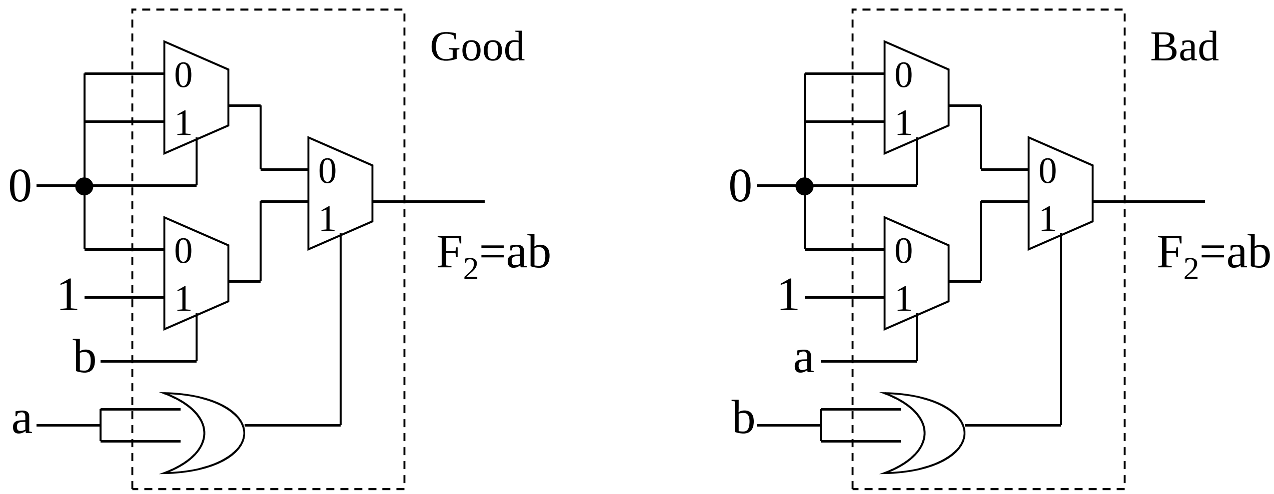
**Assume the delay of OR gate is smaller than that of MUX**

# Speed Considerations

- To reduce propagation delay, always place signals which arrive late close to the output.



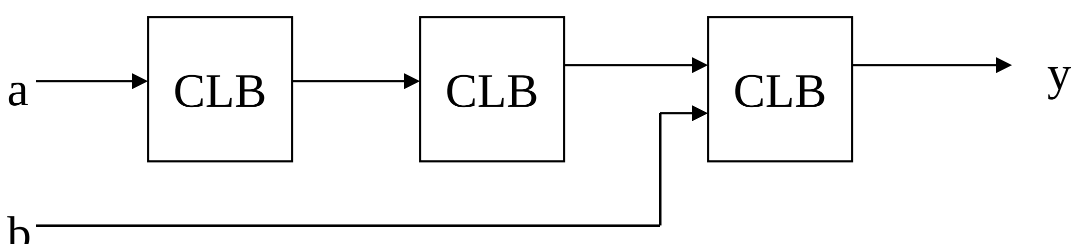

Signal e always arrives late




Signal a always arrives late

# Other Considerations

- Always try to balance signal paths to avoid glitches




If possible, change circuit implementation into the following style