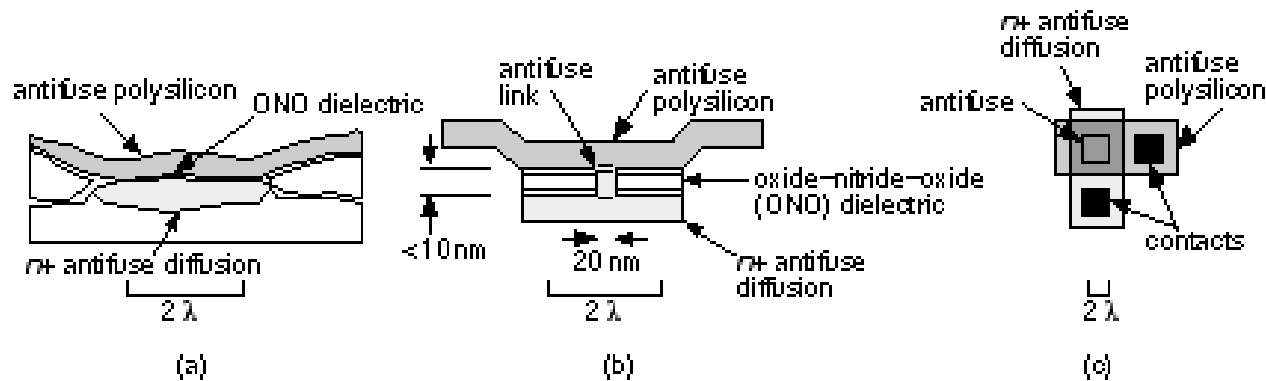


## PROGRAMMABLE ASICs

- FPGAs hold array of basic logic cells
- Basic cells configured using Programming Technologies
- Programming Technology determines basic cell and interconnect scheme
- Programming Technologies discussed:
  - Antifuse
  - SRAM
  - EPROM

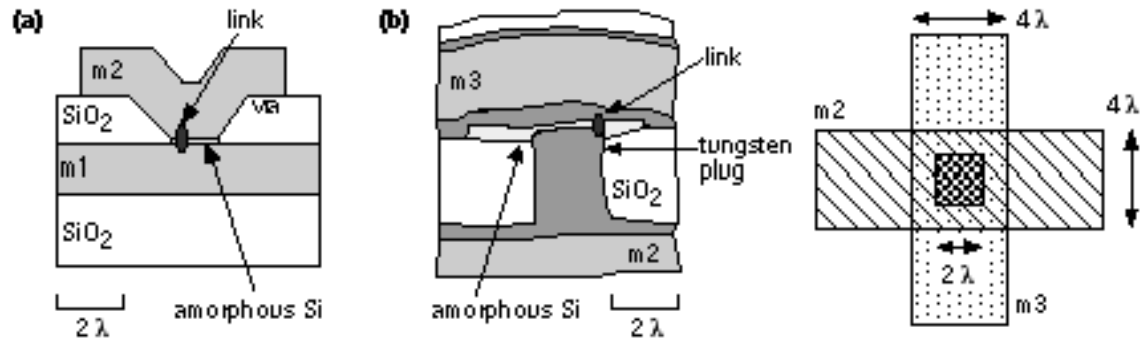
# Antifuse



## Actel antifuse

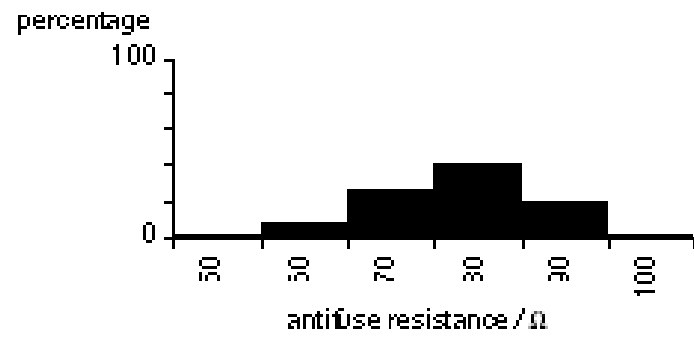
- programming current (about 5 mA)
- ( PLICE )
- oxide–nitride–oxide ( ONO ) dielectric
- Activator
- in-system programming ( ISP )
- gang programmers
- one-time programmable (OTP) FPGAs

# Metal–Metal Antifuse

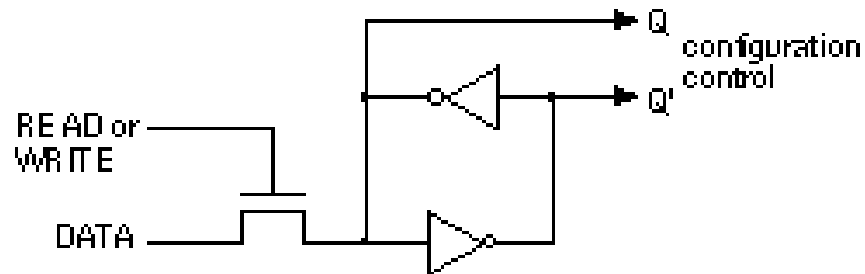


- QuickLogic metal–metal antifuse ( ‘ViaLink’ )
- alloy of tungsten, titanium, and silicon
- bulk resistance of about 500 m W cm

# Resistance values for the QuickLogic metal-metal antifuse



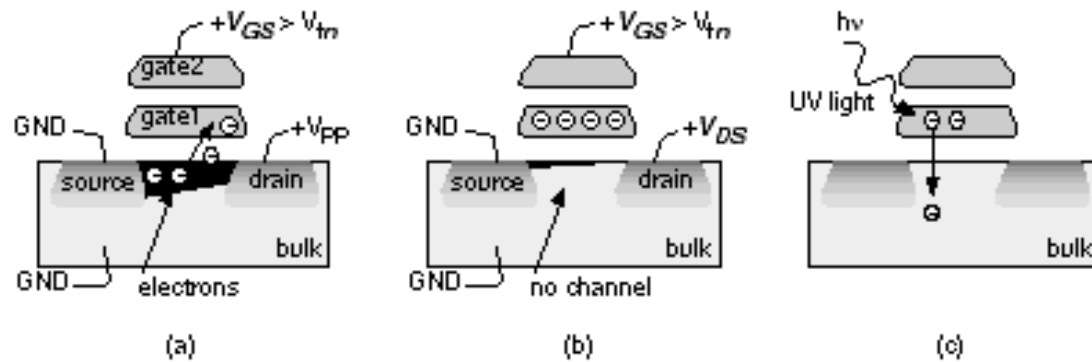
# Static RAM



## Xilinx SRAM (static RAM) configuration cell

- use in reconfigurable hardware
- use of programmable read-only memory or PROM to hold configuration

# EPROM and EEPROM Technology



An EPROM transistor

- (a) With a high ( $> 12\text{ V}$ ) programming voltage,  $V_{PP}$ , applied to the drain, electrons gain enough energy to “jump” onto the floating gate (gate1)
- (b) Electrons stuck on gate1 raise the threshold voltage so that the transistor is always off for normal operating voltages
- (c) UV light provides enough energy for the electrons stuck on gate1 to “jump” back to the bulk, allowing the transistor to operate normally

## **FPGAs in Use**

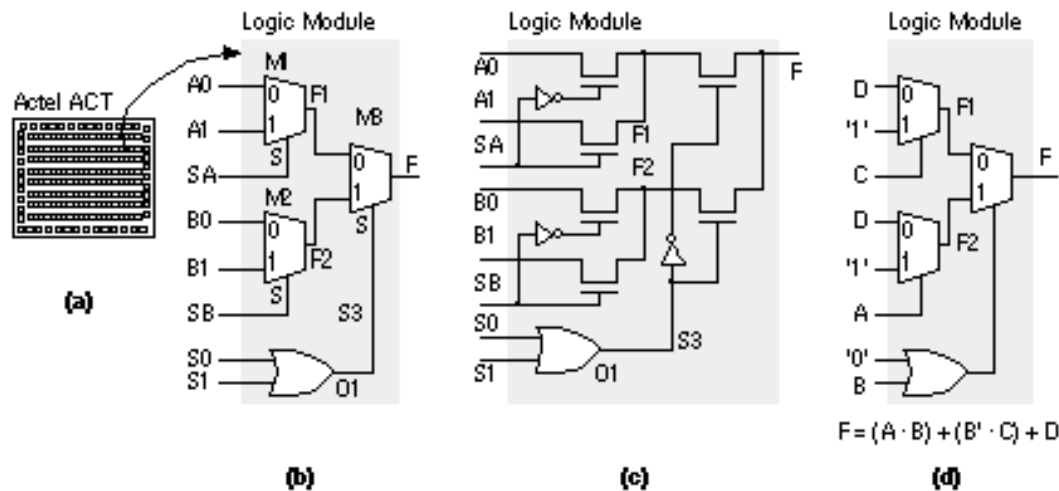
- inventory
- risk inventory or safety supply
- just-in-time ( JIT )
- printed-circuit boards ( PCBs )
- pin locking or I/O locking

# PROGRAMMABLE ASIC LOGIC CELLS

Basic logic cell can be based on

- Multiplexer
- Look-up table
- Programmable array logic

# ACT 1 Logic Module



The Actel ACT architecture

(a) Organization of the basic logic cells

(b) The ACT 1 Logic Module (LM, the Actel basic logic cell). The ACT 1 family uses just one type of LM. ACT 2 and ACT 3 FPGA families both use two different types of LM

(c) An example LM implementation using pass transistors (without any buffering)

(d) An example logic macro. Connect logic signals to some or all of the LM inputs, the remaining inputs to VDD or GND

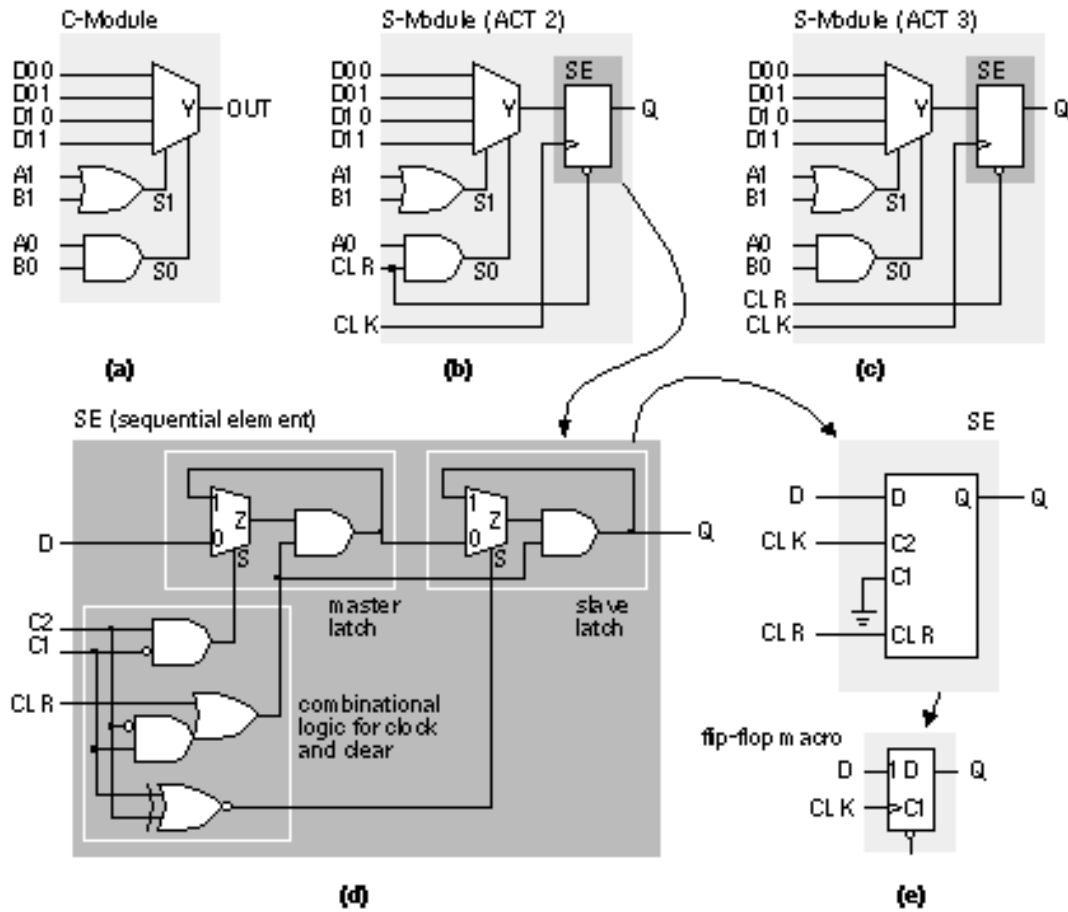
## Shannon's Expansion Theorem

- We can use the Shannon expansion theorem to expand  $F = A \cdot F(A = '1') + A' \cdot F(A = '0')$
- Example:  $F = A' \cdot B + A \cdot B \cdot C' + A' \cdot B' \cdot C = A \cdot (B \cdot C') + A' \cdot (B + B' \cdot C)$
- $F(A = '1') = B \cdot C'$  is the cofactor of  $F$  with respect to ( wrt )  $A$  or  $F_A$
- If we expand  $F$  wrt  $B$ ,  $F = A' \cdot B + A \cdot B \cdot C' + A' \cdot B' \cdot C = B \cdot (A' + A \cdot C') + B' \cdot (A' \cdot C)$
- Eventually we reach the unique canonical form , which uses only minterms
- (A minterm is a product term that contains all the variables of  $F$ —such as  $A \cdot B' \cdot C$ )

example:  $F = (A \cdot B) + (B' \cdot C) + D$

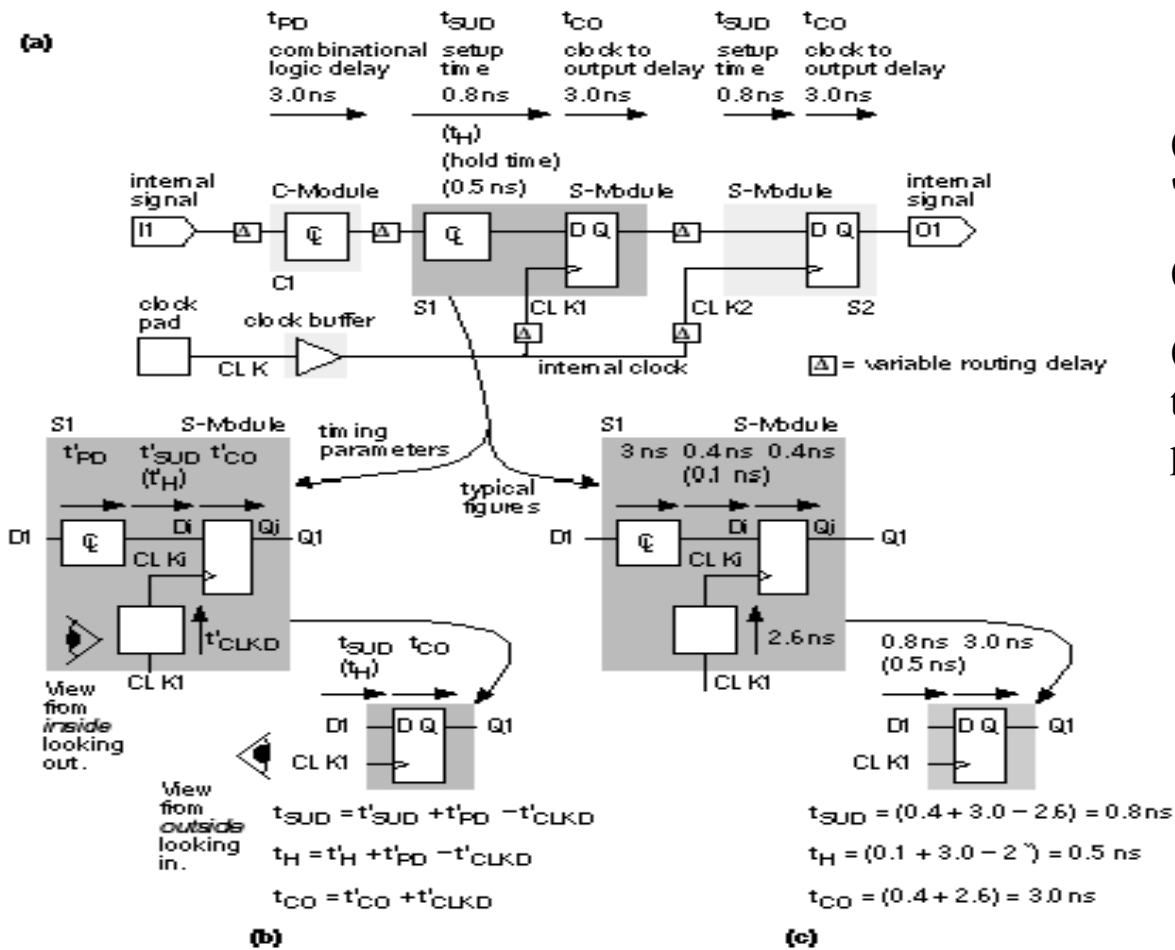
- Expand F wrt B:  $F = B \cdot (A + D) + B' \cdot (C + D) = B \cdot F2 + B' \cdot F1$
- $F = 2:1$  MUX, with B selecting between two inputs: F (A = '1') and F (A = '0')
- F also describes the output of the ACT 1 LM
- Now we need to split up F1 and F2
- Expand F2 wrt A, and F1 wrt C:  $F2 = A + D = (A \cdot 1) + (A' \cdot D)$ ;  $F1 = C + D = (C \cdot 1) + (C' \cdot D)$
- A, B, C connect to the select lines and '1' and D are the inputs of the MUXes in the ACT 1 LM
- Connections:  $A0 = D$ ,  $A1 = '1'$ ,  $B0 = D$ ,  $B1 = '1'$ ,  $SA = C$ ,  $SB = A$ ,  $S0 = '0'$ , and  $S1 = B$

# Timing Model and Critical Path



- (a) The C-Module for combinational logic
- (b) The ACT 2 S-Module
- (c) The ACT 3 S-Module
- (d) The equivalent circuit (without buffering) of the SE (sequential element)
- (e) The SE configured as a positive-edge-triggered D flip-flop

# Timing views from inside and outside the Actel ACT S-module



(a) Timing parameters for a 'Std' speed grade ACT 3

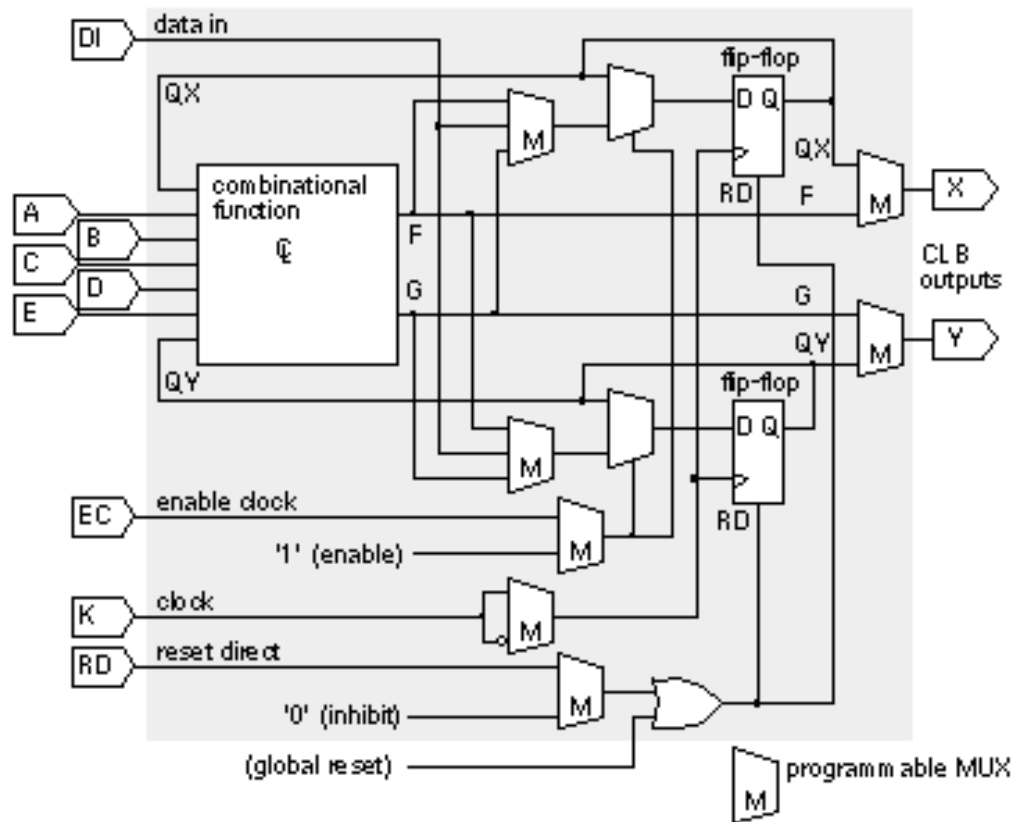
(b) Flip-flop timing

(c) An example of flip-flop timing based on ACT 3 parameters

## Actel Logic Module Analysis

- Actel uses a fine-grain architecture which allows you to use almost all of the FPGA
- Synthesis can map logic efficiently to a fine-grain architecture
- Physical symmetry simplifies place-and-route (swapping equivalent pins on opposite sides of the LM to ease routing)
- Matched to small antifuse programming technology
- LMs balance efficiency of implementation and efficiency of utilization
- A simple LM reduces performance, but allows fast and robust place-and-route

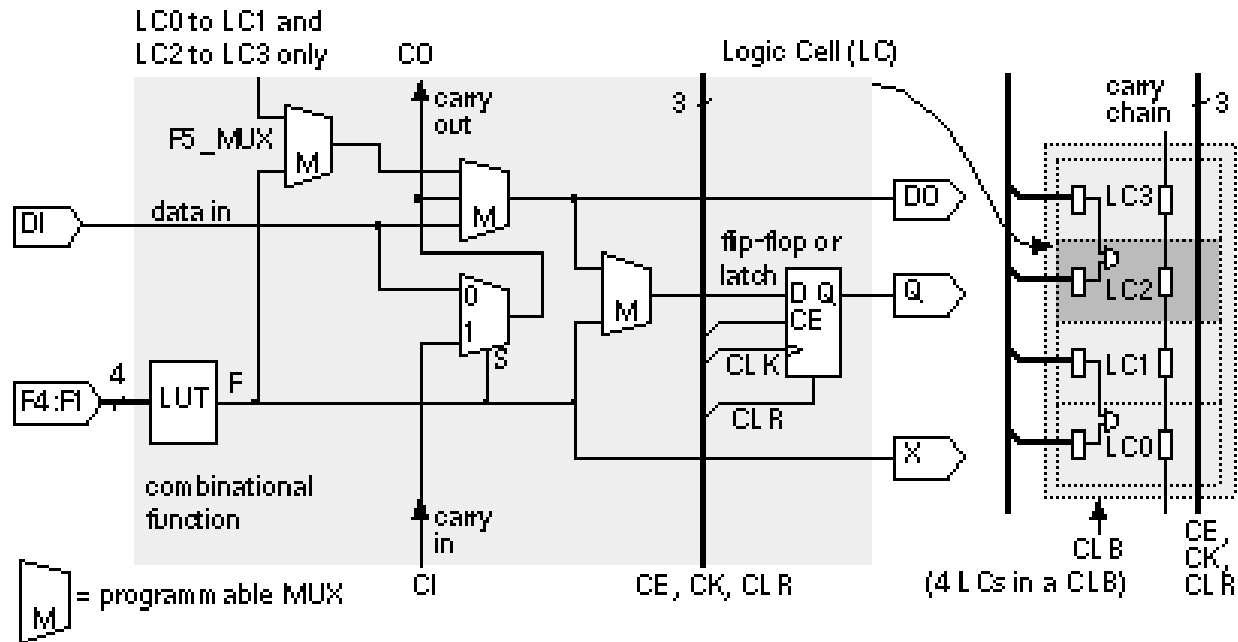
# XC3000 CLB (configurable logic block)



- A 32-bit look-up table ( LUT )
- CLB propagation delay is fixed (the LUT access time) and independent of the logic function
- 7 inputs to the XC3000 CLB: 5 CLB inputs (A–E), and 2 flip-flop outputs (QX and QY)
- 2 outputs from the LUT (F and G). Since a 32-bit LUT requires only five variables to form a unique address ( $32 = 2^5$ ), there are several ways to use the LUT:
- Use 5 of the 7 possible inputs (A–E, QX, QY) with the entire 32-bit LUT (the CLB outputs (F and G) are then identical)
- Split the 32-bit LUT in half to implement 2 functions of 4 variables each; choose 4 input variables from the 7 inputs (A–E, QX, QY). You have to choose 2 of the inputs from the 5 CLB inputs (A–E); then one function output connects to F and the other output connects to G.
- You can split the 32-bit LUT in half, using one of the 7 input variables as a select input to a 2:1 MUX that switches between F and G (to implement some functions of 6 and 7 variables).

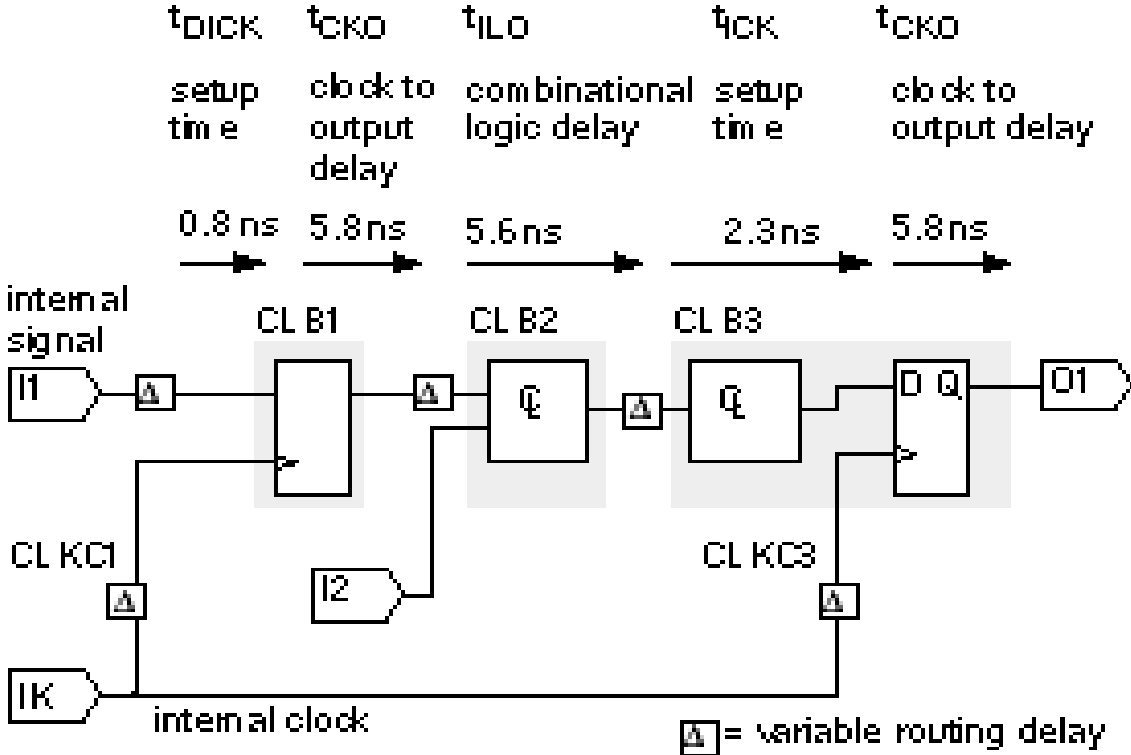


# XC5200 Logic Block



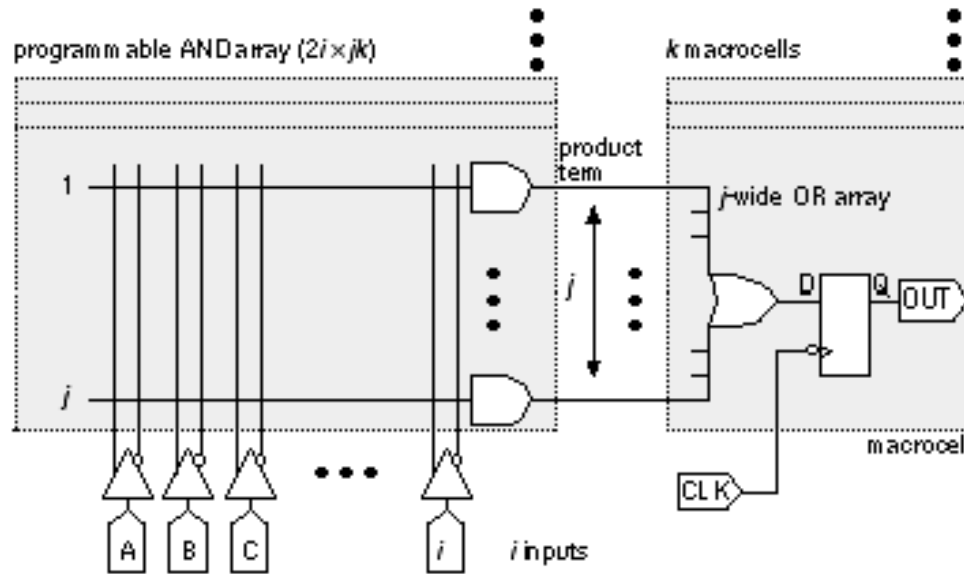
The Xilinx XC5200 family Logic Cell (LC) and configurable logic block (CLB).  
( Source: Xilinx.)

# Xilinx LCA timing model (XC5210-6)





# Altera MAX



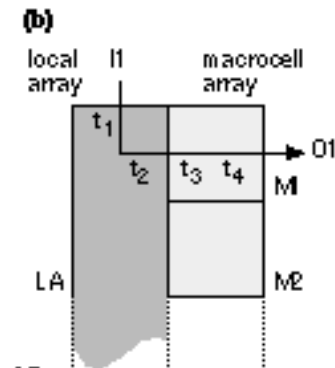
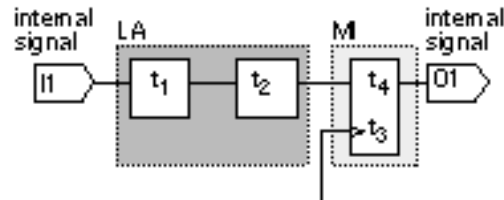
- (a) Organization of logic and interconnect
- (b) LAB (Logic Array Block)
- (c) Macrocell

## Features:

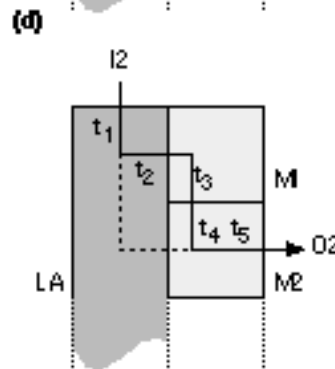
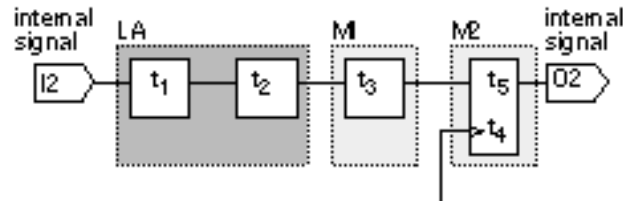
- Logic expanders and expander terms ( helper terms) increase term efficiency
- Shared logic expander ( shared expander, intranet) and parallel expander (internet)
- Deterministic architecture allows deterministic timing before logic assignment
- Any use of two-pass logic breaks deterministic timing
- Programmable inversion increases term efficiency

# Timing Model - Altera MAX

(a)  $t_{LOCAL}$   $t_{LAD}$   $t_{SU}$   $t_{RD}$   
 local logic setup register  
 array array delay delay  
 0.5 4.0 3.0 1.0 total = 8.5 ns



(c)  $t_{LOCAL}$   $t_{LAD}$   $t_{EXP}$   $t_{SU}$   $t_{RD}$   
 local logic parallel setup register  
 array array expander delay delay  
 0.5 4.0 1.0 3.0 1.0 total = 9.5 ns



(e)  $t_{LOCAL}$   $t_{LAD}$   $t_{SEXP}$   $t_{LOCAL}$   $t_{COMB}$   
 local logic shared local combinational  
 array array expander array delay  
 0.5 4.0 5.0 0.5 1.0 total = 11 ns

