MODULE- 3

Programmable ASICs: The Antifuse, EPROM and EEPROM Technology, Practical Issues, Specifications (4.1,4.2,4.3,4.4 & 4.5)

Programmable ASICs logic cells: Actel ACT, Xilinx LCA, Altera FLEX, Altera MAX (5.1,5.2,5.3 & 5.4)

Programmable ASIC I/O cells: DC output, AC output, DC input, AC input, Clock input, Power input, Xilinx I/O Block, other I/O cells. (6.1,6.2,6.3,6.4,6.5,6.6,6.7 & 6.8)

by,

Dr.P.Vimala
Department of ECE
Dayananda Sagar College of Engineering
Bangalore

Figures from M.J.S .Smith, - "**Application - Specific Integrated Circuits**"

Programmable ASICs

- Two basic types of programmable ASICs
  - Programmable Logic Device (PLD) - first developed as small programmable devices least complex ones are a simple AND/OR PLA with latches on the outputs
  - Field Programmable Gate Array (FPGA) - more complex devices that can hold up to 100K gate equivalents or more
- An FPGA is a chip- As a system designer can program themselves. An IC foundry produces FPGA with some connections missing.
- The user (designer) creates a design to be placed on the FPGA using design entry and simulation
- Automatic tools create a string of bits (a *configuration file*) describing the extra connections necessary to program the FPGA to perform the required function
- A *device programmer* is then (usually) used to load the configuration file into the FPGA

FPGA Components

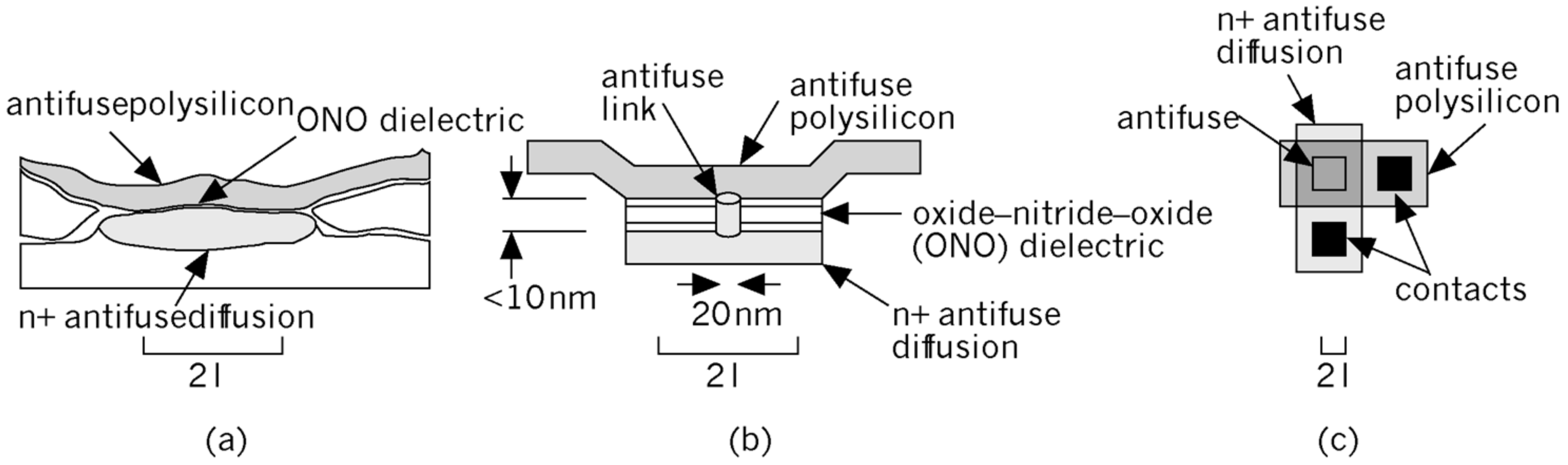
- FPGAs have several basic components:
  - Regular array of basic (programmable) logic cells
  - Programmable interconnect for connecting the basic cells into different configurations
  - Programming technology for configuring the cells and programmable interconnect
    - One-time-programmable (OTP)
    - Erasable
    - Programmed on power-up
  - Custom software used by the designer to create the configuration file

Programming Technology

- The designer uses custom software, **tailored to each programming technology and FPGA architecture**, to design and implement the programmable connections.
- Programming technology in an FPGA **determines the type of basic logic cell and the interconnect scheme.**
- The logic cells and interconnection scheme **determine the design of the input and output circuits as well as the programming scheme.**
- **The programming technology may or may not be permanent.**
  - Cannot undo the permanent programming in one-time programmable (OTP) FPGAs.
  - Reprogrammable or erasable devices may be reused many times.

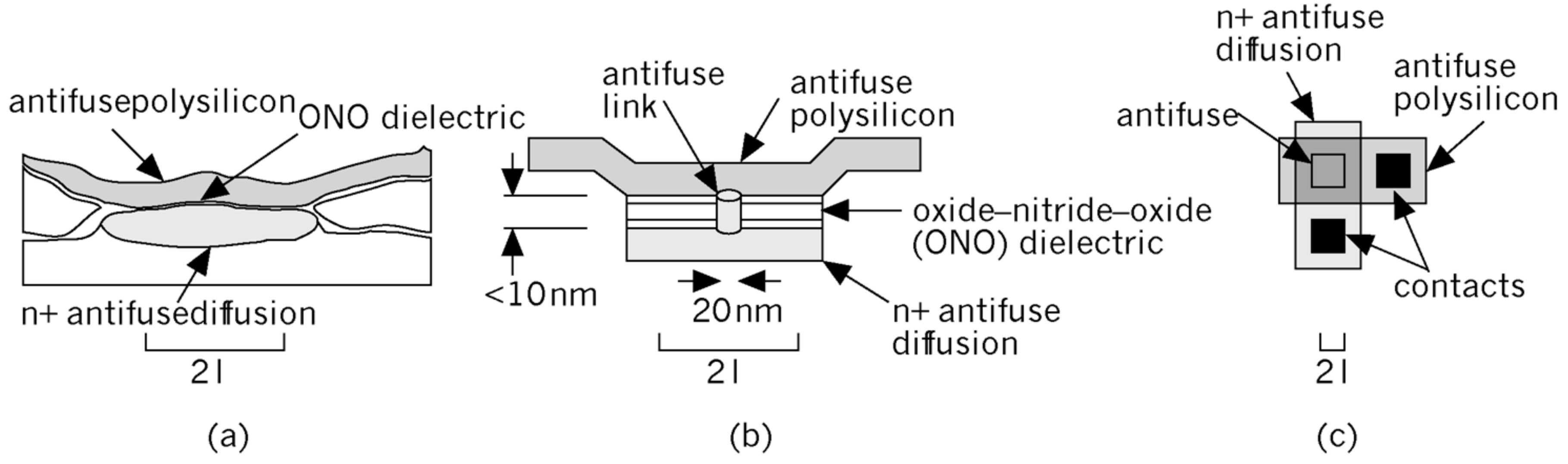
The Antifuse

- An antifuse is normally open
- A high programming voltage is placed across it
- This forces a programming current (about 5 mA) through it which melts the thin insulating dielectric forming a permanent, resistive silicon link
- In a poly diffusion antifuse the high current density causes a large power dissipation in a small area, which melts a thin insulating dielectric between polysilicon and diffusion electrodes and forms a thin (about 20 nm in diameter), permanent, and resistive silicon link .



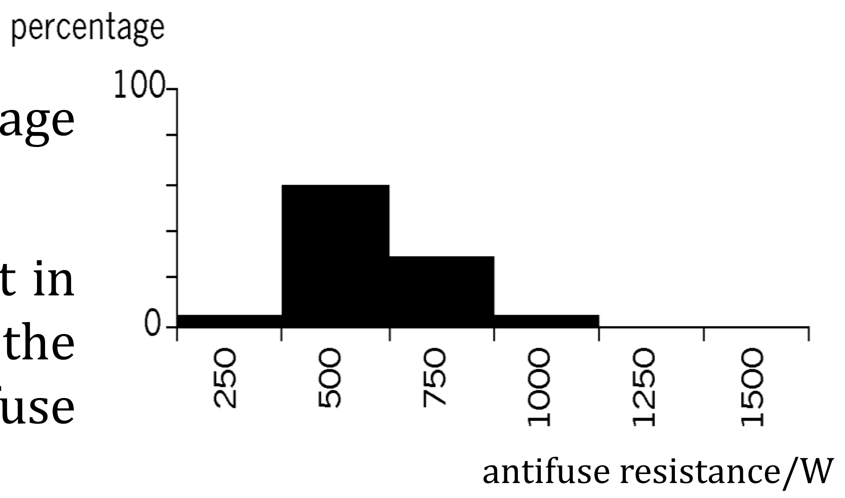
An Actel antifuse. (a) A cross section. (b) A simplified drawing. (c) From above, an antifuse is approximately the same size as a contact.

- Actel calls its antifuse a **programmable low-impedance circuit element (PLICE).**
- **P**oly–diffusion antifuse with an oxide–nitride–oxide (ONO) dielectric sandwich of: silicon dioxide (SiO_2) grown over the n -type antifuse diffusion, a silicon nitride ($Si_3 N_4$) layer, and another thin SiO_2 layer.
- The layered ONO dielectric results in a **tighter spread of blown antifuse resistance values** than using a single-oxide dielectric.
- The effective electrical thickness is equivalent to 10nm of SiO_2 ($Si_3 N_4$ has a higher dielectric constant than SiO_2 , so the actual thickness is less than 10 nm).
- Sometimes this device is called a fuse even though it is an anti fuse, and both terms are often used interchangeably.



Actel Antifuses

- The fabrication process and the programming current control the average resistance of a blown antifuse, but values vary as shown in Figure.
- In a particular technology a programming current of 5 mA may result in an average blown antifuse resistance of about 500 W . Increasing the programming current to 15 mA might reduce the average antifuse resistance to 100 W .
- Antifuses separate interconnect wires on the FPGA chip and the programmer blows an antifuse to make a permanent connection. Once an antifuse is programmed, the process cannot be reversed. **This is an one time programming (OTP) technology** (and radiation hard).
- An Actel 1010, for example, contains 112,000 antifuse (Table 4.1).
- Actel antifuse technology uses three additional masks over a traditional CMOS process
- Programming an ACTEL device requires about 5 to 10 minutes per device
- Production programming of more than 1000 or 2000 devices per week requires a *gang* (multiple device) *programmer*



Distribution of resistances for blown Actel antifuses.

Table 4.1 Number of antifuses on Actel FPGAs

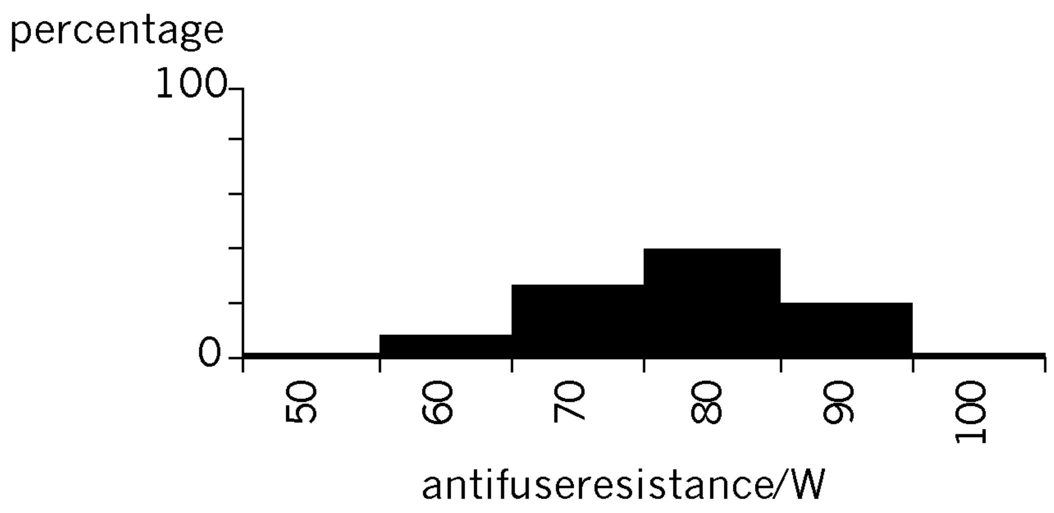
Device	Antifuses
A1010	112,000
A1020	186,000
A1225	250,000
A1240	400,000
A1280	750,000

- To design and program an Actel FPGA, designers iterate between design entry and simulation.
- When they are satisfied the design is correct they **plug the chip into a socket** on a special programming box, called **an Activator** , that generates the programming voltage.
- A PC downloads the configuration file to the Activator instructing it to blow the necessary antifuse on the chip.
- When the chip is programmed it may be removed from the Activator without harming the configuration data and the chip assembled into a system.
- One **disadvantage** of this procedure is that modern packages with **hundreds of thin metal leads are susceptible to damage** when they are inserted and removed from sockets.
- The advantage of **other programming technologies** is that **chips may be programmed after they have been assembled on a printed-circuit board**—a feature known as **in-system programming (ISP)**

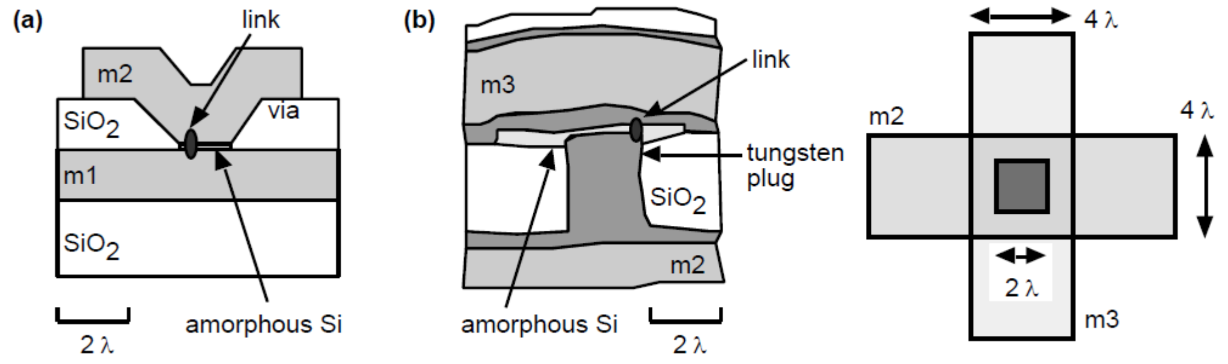
Metal-Metal Antifuse

- QuickLogic metal–metal antifuse (ViaLink ‘)
- The link is an alloy of tungsten, titanium, and silicon with a bulk resistance of about 500 mW cm.
- Two Advantages :
  1. Metal-metal Antifuses directly connect metal wiring layers - thus eliminating the parasitic of a polysilicon layer in between
  2. Direct connections to the metal layers make it easier to use larger programming currents producing a lower antifuse resistance

Metal-Metal Antifuse Resistance



Distribution of resistance values for the Quick Logic metal-metal antifuse.

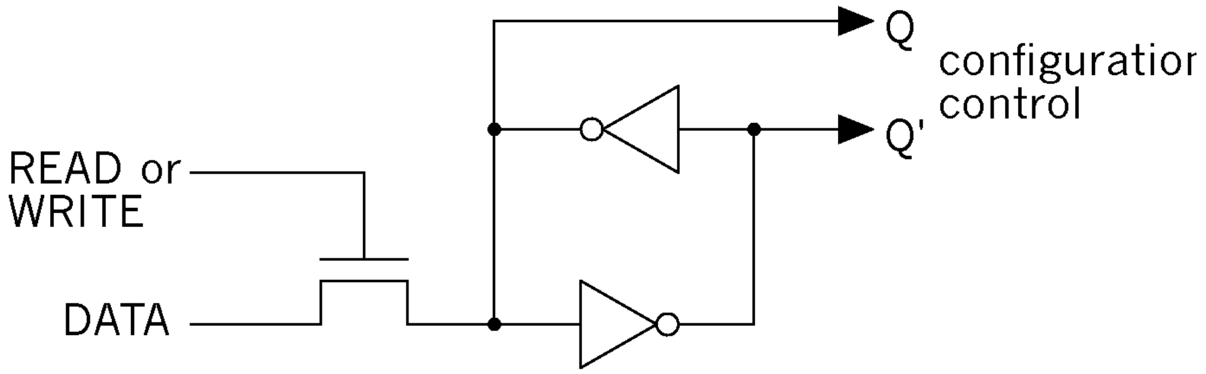


Metal-metal antifuse. (a) An idealized cross section. (b) A metal-metal antifuse in a three-level metal process.

Static RAM

Xilinx SRAM (static RAM) configuration cell,

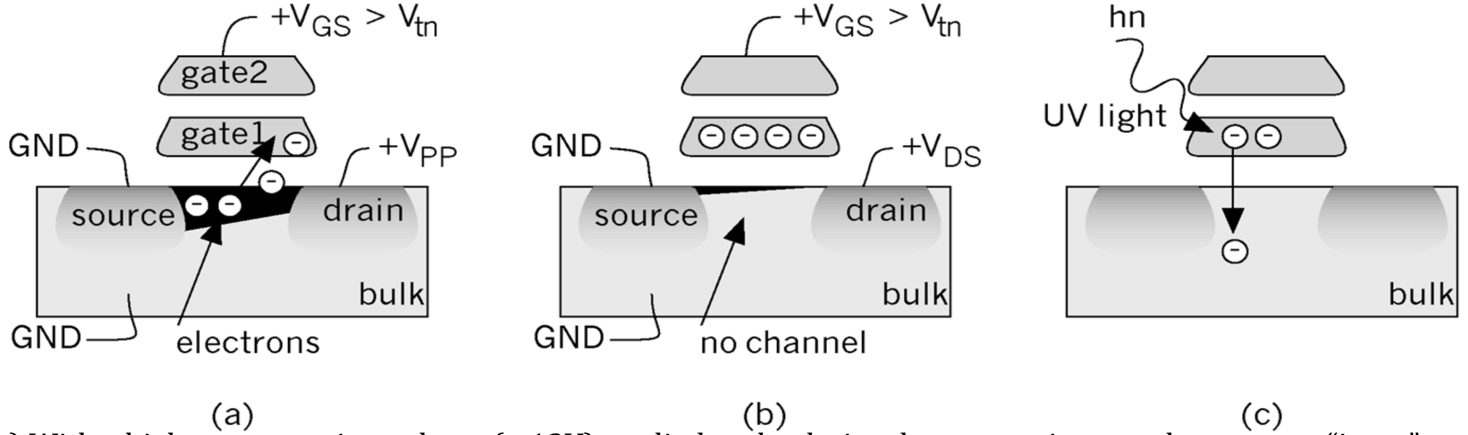
- Use in reconfigurable hardware
- Use of programmable read-only memory or PROM to hold configuration
- Power must be maintained to the chip to retain the configuration or the configuration can be loaded from a PROM on power-up
- **The advantages of SRAM** programming technology are that designers can **reuse chips** during prototyping
- The **disadvantage** of using SRAM programming technology is that you need to **keep power supplied to the programmable ASIC** (at a low level) for the volatile SRAM to retain the connection information.



The Xilinx SRAM(static RAM) configuration cell. The outputs of the cross-coupled inverter (configuration control)are connected to the gates of pass transistors or transmission gates. The cell is programmed using the WRITE and DATA lines.

EPROM Cell

- The EPROM cell is almost as small as an antifuse.
- **Altera MAX 5000 EPLDs and Xilinx EPLDs** both use UV-erasable electrically programmable read-only memory (EPROM)
- An EPROM transistor looks like a normal MOS transistor except it has a second, floating, gate (gate1 in Fig).
- Applying a programming voltage V_{PP} (usually greater than 12 V) to the drain of the n-channel EPROM transistor programs the EPROM cell.
- A high electric field causes electrons flowing toward the drain to move so fast they jump transistor programs the EPROM cell across the insulating gate oxide where they are trapped on the bottom, floating, gate.
- These energetic electrons are hot and the effect is known as **hot-electron injection or avalanche injection** . EPROM technology is sometimes called **floating-gate avalanche MOS (FAMOS).**



An EPROM transistor. (a) With a high programming voltage (> 12V) applied to the drain, electrons gain enough energy to “jump” onto the floating gate. (b) Electrons stuck on gate 1 raise the threshold voltage so that the transistor is always off for normal operating conditions. (c) UV light provides enough energy to the stuck electrons on gate 1 for them to “jump” back to the bulk.

Practical Issues

- Most computer-aided engineering (CAE) software for FPGA design uses some type of security.
- For workstations this usually means **floating licenses** (any of n-users on a network can use the tools) **or node-locked licenses** (only n particular computers can use the tools) **using the hostid** (or host I.D., a serial number unique to each computer) in the boot EPROM .
- For PCs this is a hardware key.
- Some keys use the serial port (requiring extra cables and adapters); and some use the parallel port.
- There are often conflicts between keys and other hardware/software.

•For example, for a while some security keys did not work with the serial-port driver on Intel motherboards—users had to buy another serial-port I/O card.

There are many other factors to be considered in choosing hardware:

- Software packages are normally less expensive on a PC.
- Peripherals are less expensive and easier to configure on a PC.
- Maintenance contracts are usually necessary and expensive for workstations.
- There is a much larger network of users to provide support for PC users.
- It is easier to upgrade a PC than a workstation

Programmable ASICs logic cells (5.1,5.2,5.3 & 5.4)

Actel ACT
Xilinx LCA
Altera FLEX
Altera MAX

Programmable ASICs logic cells

- All FPGAs contain a basic logic cell replicated in a regular array across the chip.
- There are three different types of basic logic cells:
  - multiplexer based
  - look-up table based
  - programmable array based
- The choice among these depends on the programming technology.

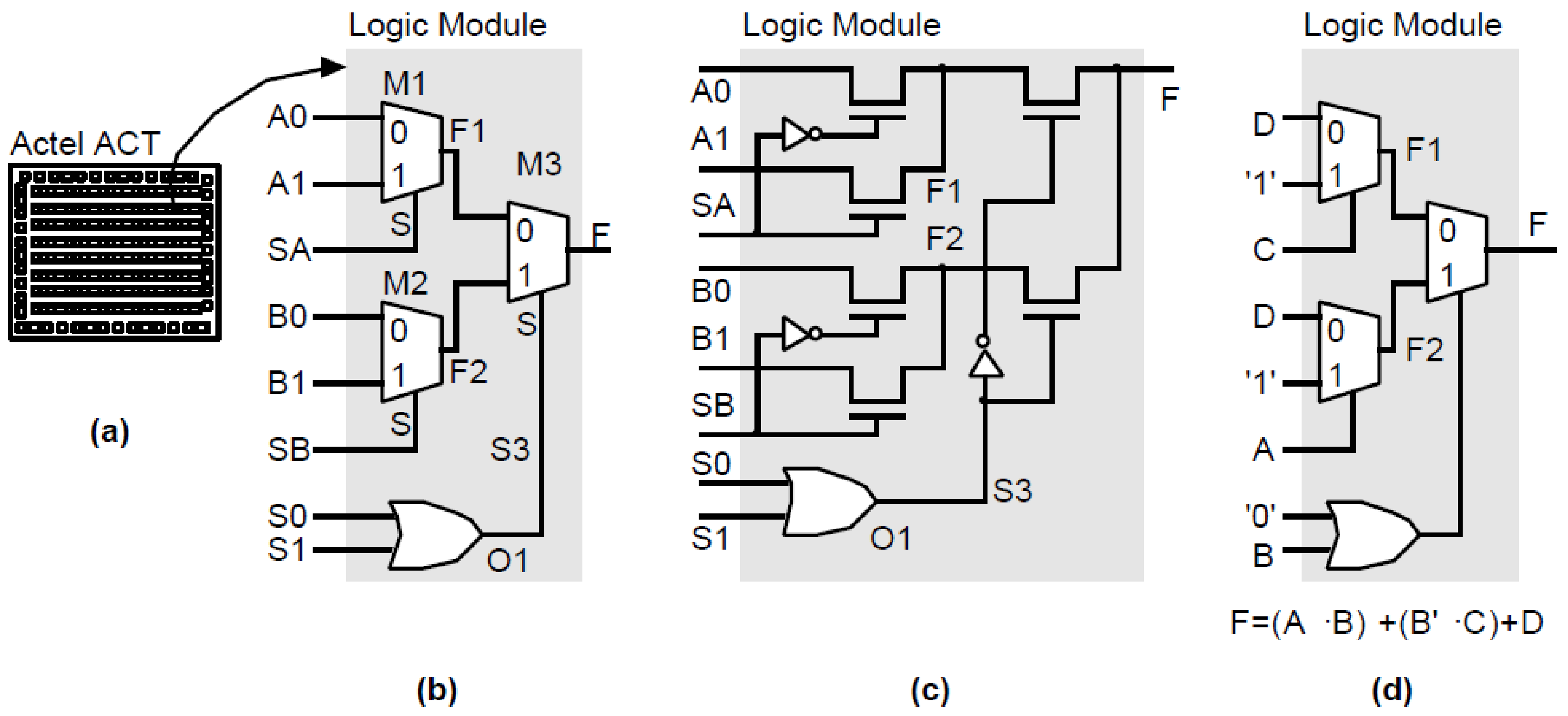
Actel ACT

- The basic logic cells in the Actel ACT family of FPGAs are called **Logic Modules .**
- ACT-1 family uses just one type of Logic Module
- ACT-2and ACT-3 FPGA families both use two different types of Logic Module.

Actel ACT1 :

Multiplexer Based Logic Cell

- Logic functions can be built by connecting logic signals to some or all of the Logic Module's inputs and by connecting the remaining Logic Module inputs to VDD or GND



The Actel ACT architecture:
(a) Organization of the basic logic cells
(b) The ACT 1 Logic Module (LM, the Actel basic logic cell).
(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 the function,

F =A·F(A='1’) + A'·F(A='0')

Example1: F =A'·B + A·B·C' + A'·B'·C = A·(B·C') + A'·(B + B'·C)

- F(A='1')=B·C' is the **cofactor** of F with respect to (wrt) A or FA
- If we expand F wrt B, F =A'·B + A·B·C' + A'·B'·C = B·(A' + A·C') + B'·(A'·C)
- Eventually we reach the unique **canonical form,** which uses only minterms
- F = 2:1 MUX, with B selecting between two inputs: F(B='1') and F(B='0')

Using Shannon's Expansion Theorem to Map a Function to an ACT1 Logic Module

- **example2:** F = (A · B) + (B' · C) + D

  Expand F wrt B: F = B · (A + D) + B' · (C + D) = B · F2 + B' · F1

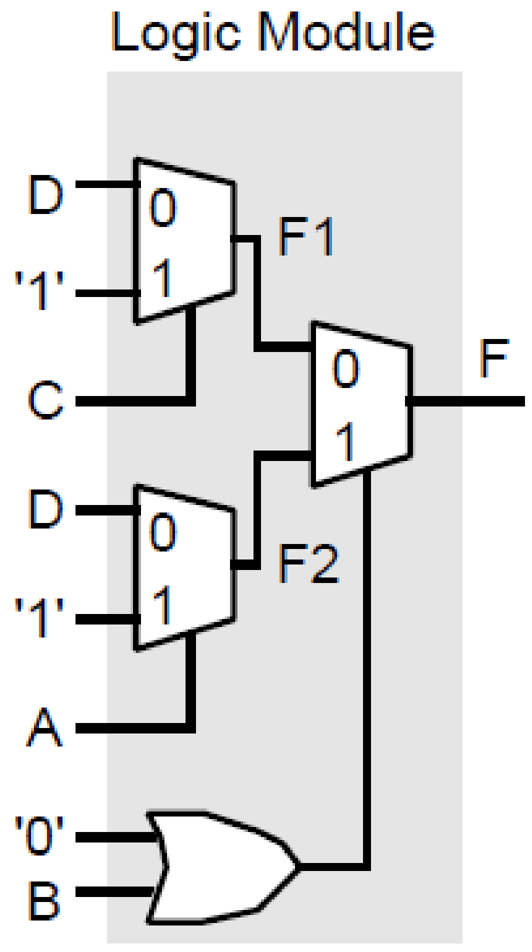
  Where F1= (C + D) and F2 = (A + D)

- The function F can be implemented by 2:1 MUX, with B selecting between two (B = '0')
  - F also describes the output of the ACT 1 LM
- Now we need to split up F1 and F2

  Expand F1 wrt C: F1 = C + D = (C · 1) + (C' · D)

  Expand F2 wrt A: F2 = A + D = (A · 1) + (A' · D);

- **C connects to the select line of a first-level mux** in the ACT1 LM with '1' and mux
- **A connects to the select line of another first-level mux** in the ACT1 LM with the mux
- **B connects to the select line of the output mux** with F1 and F2, the outputs of the first level muxes, connected to the inputs

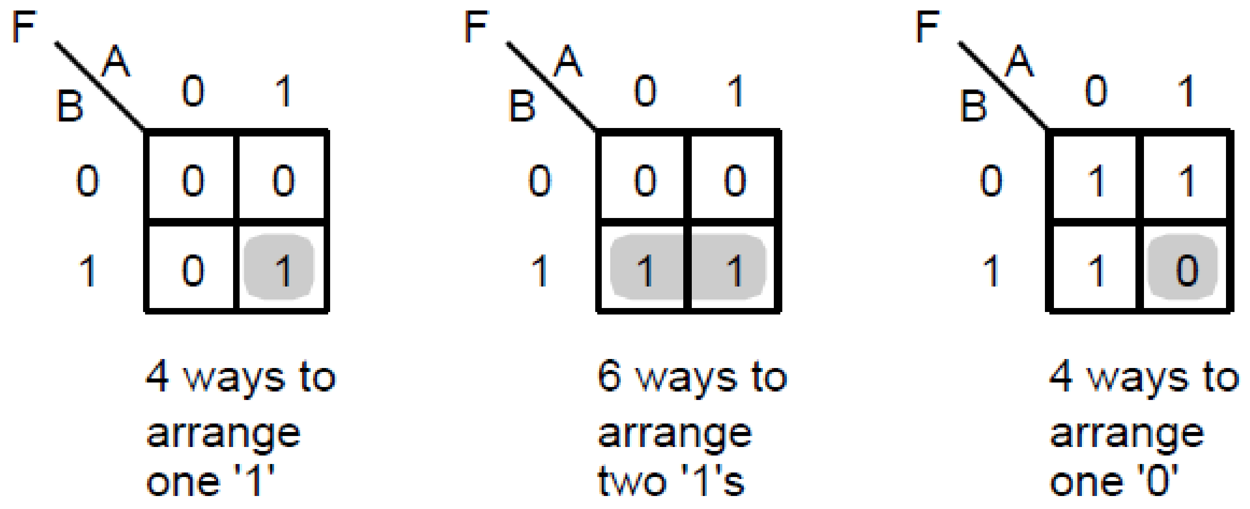


F=(A ·B) +(B' ·C)+D

Multiplexer Logic as Function Generators

The 16 logic functions of 2 variables:

- 2 of the 16 functions are not very interesting (F='0', and F='1’)
- There are 10 functions that we can implement using just one 2:1 MUX
- 6 functions are useful: INV, BUF, AND, OR, AND1-1, NOR1-1


F
A
B
0
1
4 ways to arrange one '1'
6 ways to arrange two '1's
4 ways to arrange one '0'

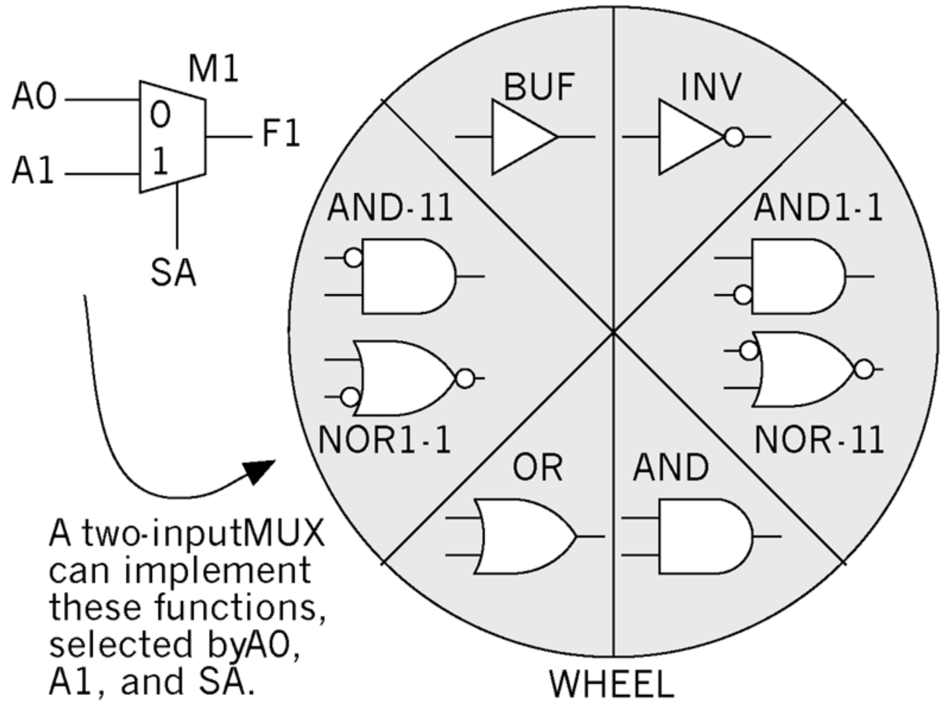
14 functions of 2 variables (and F='0', F ='1' makes 16)

Boolean Functions of Two Variables Using a 2:1 Mux

	Function, F	F =	Canonical form	Minterms	M1		
					A0	A1	SA
1	'0'	'0'	'0'	none	0	0	0
2	NOR1-1(A, B)	(A + B')	A' · B	1	B	0	A
3	NOT(A)	A'	A' · B' + A' · B	0, 1	0	1	A
4	AND1-1(A, B)	A · B'	A · B'	2	A	0	B
5	NOT(B)	B'	A' · B' + A · B'	0, 2	0	1	B
6	BUF(B)	B	A' · B + A · B	1, 3	0	B	1
7	AND(A, B)	A · B	A · B	3	0	B	A
8	BUF(A)	A	A · B' + A · B	2, 3	0	A	1
9	OR(A, B)	A + B	A' · B + A · B' + A · B	1, 2, 3	B	1	A
10	'1'	'1'	A' · B' + A' · B + A · B' + A · B	0, 1, 2, 3	1	1	1

ACT1 LM as a Function Wheel

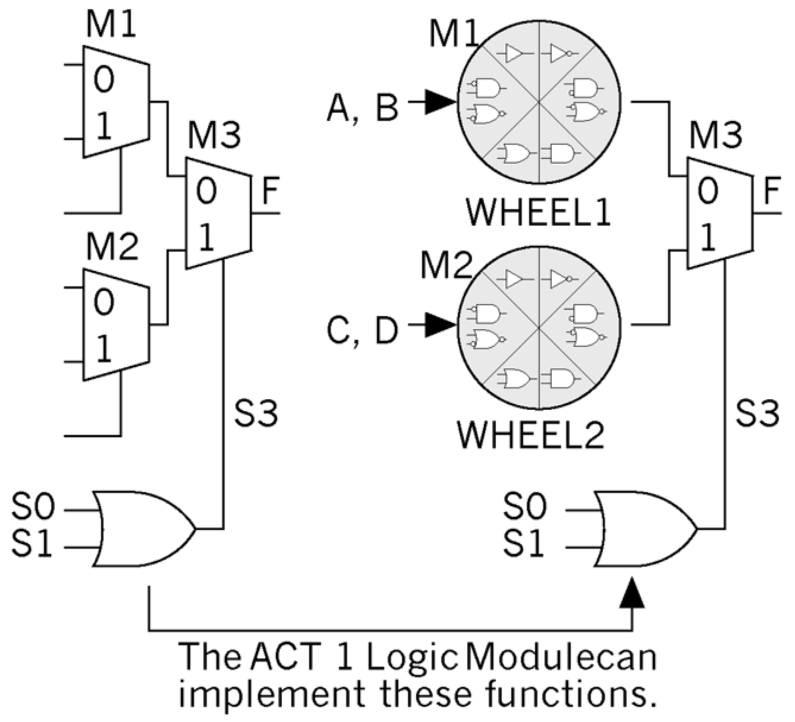
- A 2:1 MUX is a function wheel that can generate BUF, INV, AND-11, AND1-1, OR, AND
- Define a function WHEEL (A, B) = MUX (A0, A1, SA)
- MUX (A0, A1, SA) = A0 · SA' + A1 · SA
- Each of the inputs (A0, A1, and SA) may be A, B, '0', or '1'


M1
A0
A1
0
1
F1
SA
A two-inputMUX can implement these functions, selected byA0, A1, and SA.
BUF
INV
AND-11
AND1-1
NOR1-1
NOR-11
OR
AND
WHEEL

ACT1 LM as a Function Wheel

The ACT 1 LM is built from two function wheels,
a 2:1 MUX, and a two-input OR gate:

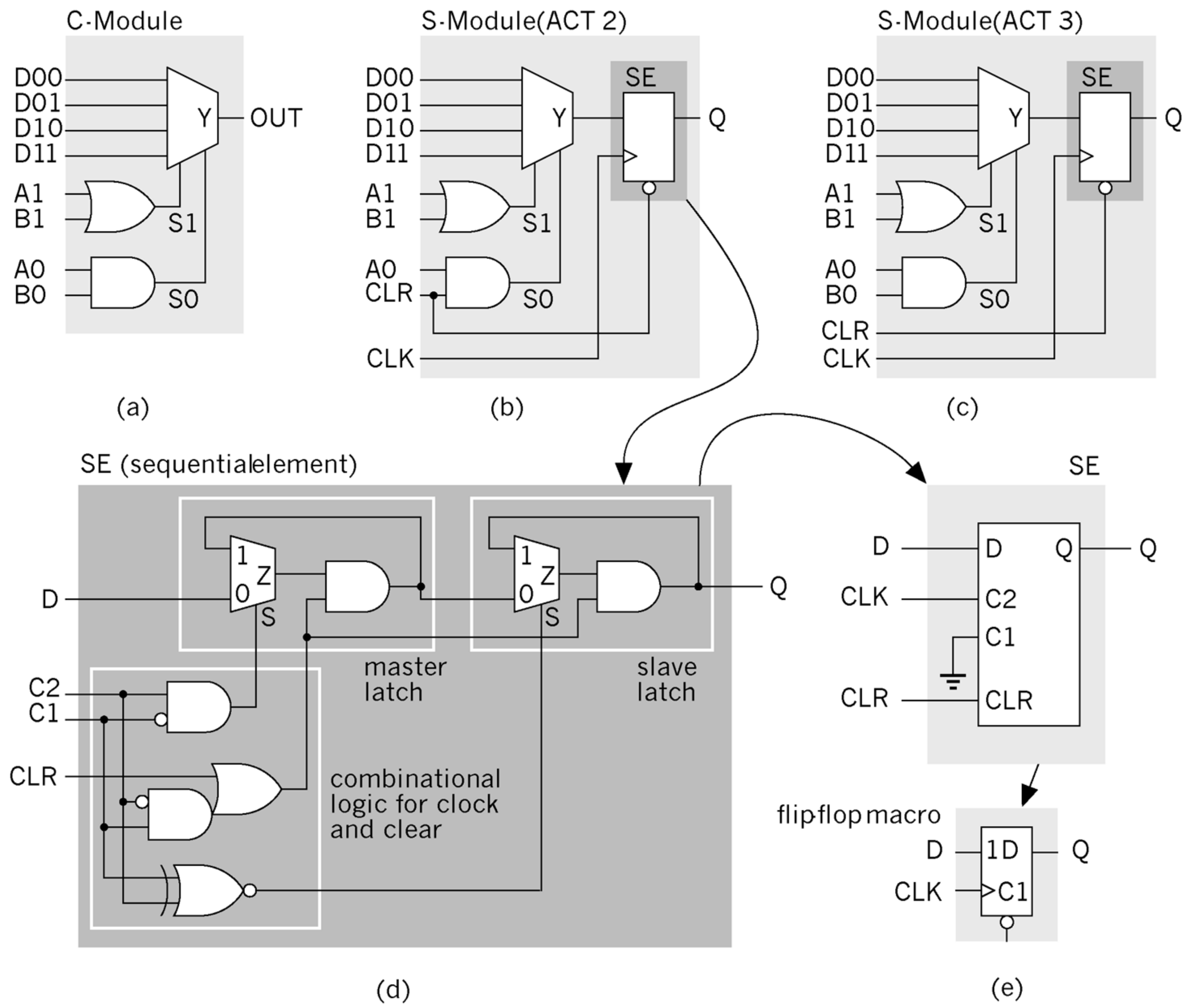
ACT 1 LM = MUX [WHEEL1, WHEEL2, OR (S0, S1)]



(b)

Example of using the WHEEL functions to implement F=NAND(A, B)=(A·B)'
1. First express F as the output of a 2:1 MUX:
we do this by expanding F wrt A (or wrt B; since F is symmetric) F=A·(B') + A'·('1')
2. Assign WHEEL1 to implement INV(B), and WHEEL2 to implement '1'
3. Set the select input to the MUX connecting WHEEL1 and WHEEL2, S0+S1=A. We can do this using S0=A, S1='1'

Actel ACT2 and ACT3 Logic Modules



- ACT 2 and ACT 3 use two types of LMs, one includes a D flip-flop
- ACT 2 C-Module is similar to the ACT 1 LM but can implement five-input logic functions
- ACT 2 S-Module (sequential module) contains a C-Module and a sequential element

The ACT2 and ACT3 logic modules.
(a)The C-module.
(b)The ACT2 S-module.
(c)The ACT3 S-module.
(d)The equivalent circuit of the SE.
(e)The SE configured as a positive edge-triggered D flip-flop.

Actel Timing Model

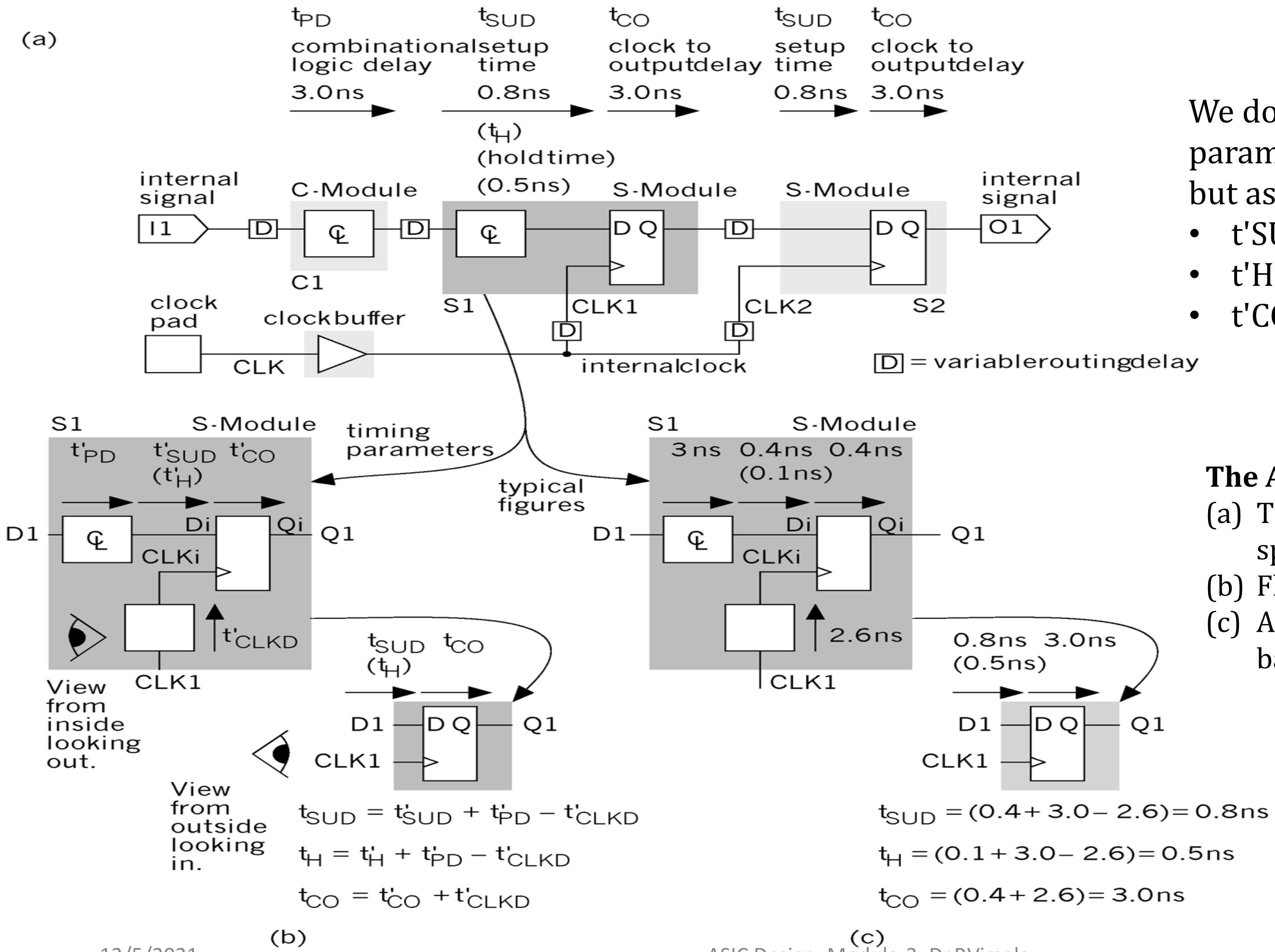
- Exact delay values in Actel FPGAs can not be determined until interconnect delay is known - i.e., place and route are done
- Critical path delay between registers is:

$$t_{PD} + t_{SUD} + t_{CO}$$

- There is also a hold time for the flip-flops - t_H
- The combinational logic delay t_{PD} is dependent on the logic function (which may take more than one LM) and the wiring delays
- The flip-flop output delay t_{CO} can also be influenced by the number of gates it drives (fan-out)

- The setup and hold times, measured *inside* (not outside) the S-Module, are t'_{SUD} and t'_{H} (a prime denotes parameters that are measured inside the S-Module)
- The clock–Q propagation delay is t'_{CO}
- The parameters t'_{SUD}, t'_{H}, and t'_{CO} are measured using the *internal* clock signal,CLKi
- The propagation delay of the combinational logic *inside* the S-Module is t'_{PD}
- The delay of the combinational logic that drives the flip-flop clock signal is t'_{CLKD}
- From *outside* the S-Module, with reference to the outside clock signal CLK1:

$t_{SUD}=t'_{SUD} + (t'_{PD} – t'_{CLKD})$; $t_{H}=t'_{H} + (t'_{PD} – t'_{CLKD})$; $t_{CO}=t'_{CO} + t'_{CLKD}$



$t_{SUD} = t'_{SUD} + t'_{PD} - t'_{CLKD}$

$t_H = t'_H + t'_{PD} - t'_{CLKD}$

$t_{CO} = t'_{CO} + t'_{CLKD}$

$t_{SUD} = (0.4 + 3.0 - 2.6) = 0.8\text{ns}$

$t_H = (0.1 + 3.0 - 2.6) = 0.5\text{ns}$

$t_{CO} = (0.4 + 2.6) = 3.0\text{ns}$

We do not know the *internal* parameters t'SUD, t'H, and t'CO, but assume reasonable values:

- t'SUD=0.4ns
- t'H=0.1ns
- t'CO=0.4ns.

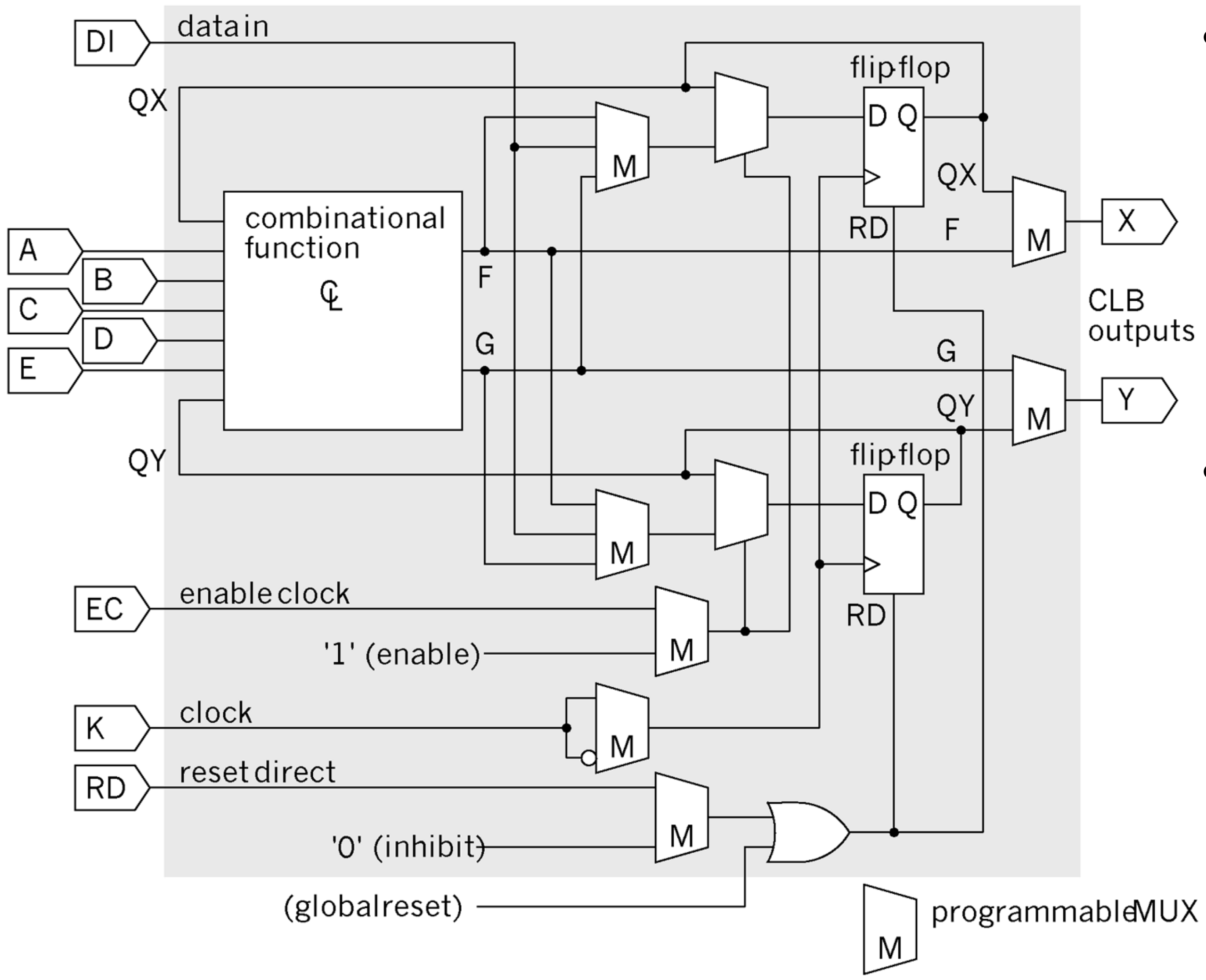
The Actel ACT timing model.

(a) The timing parameters for a ‘std’ speed grade ACT3.
(b) Flip-flop timing.
(c) An example of flip-flop timing based on ACT3 parameters.

Xilinx LCA

- Xilinx LCA basic logic cells, **configurable logic blocks (CLBs**), are bigger and more complex than the Actel
- The Xilinx CLBs contain both combinational logic and flip-flops

Xilinx XC3000 Logic Block



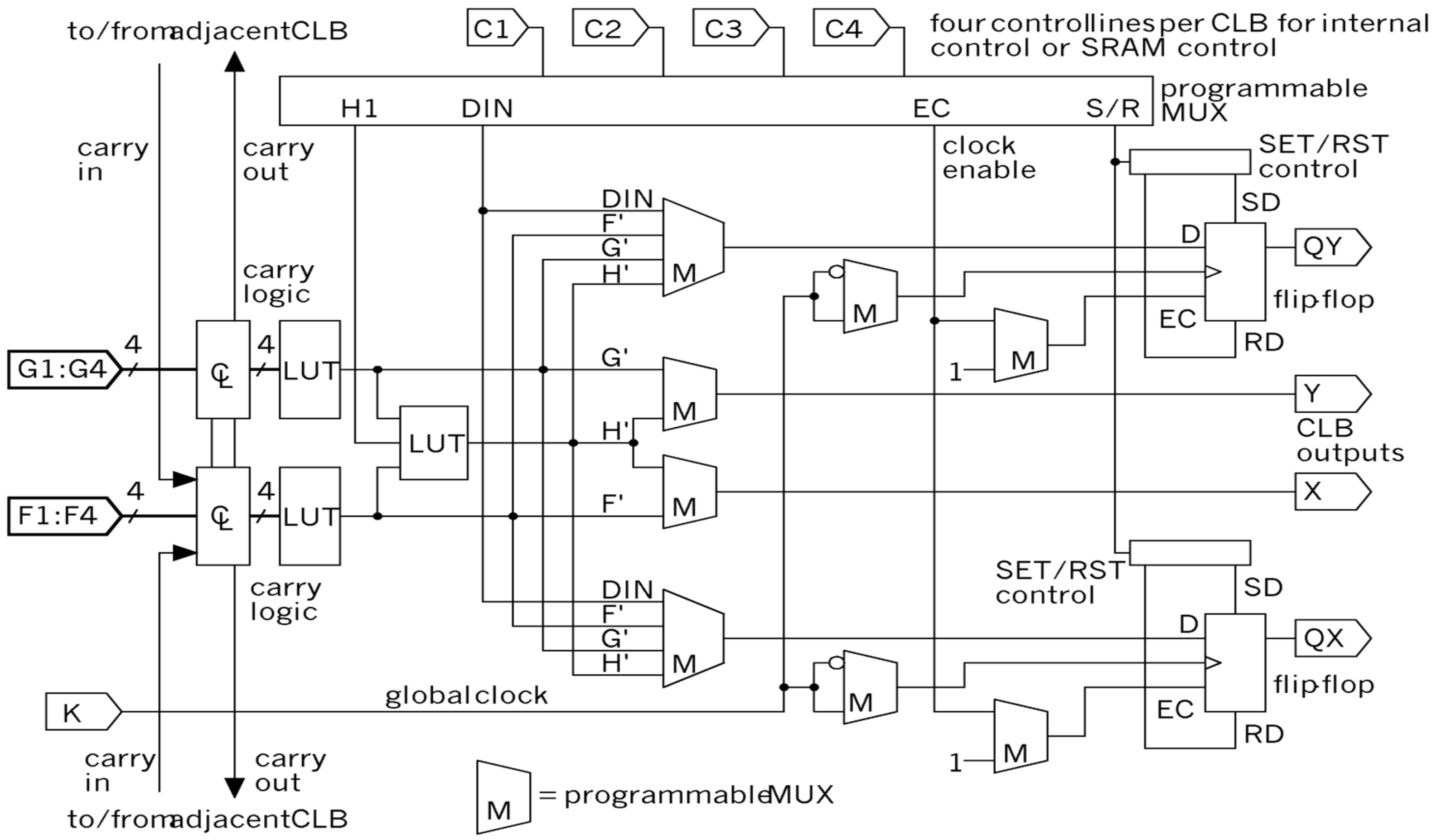
The Xilinx XC3000 CLB (configurable logic block).

- CLB I/O pins:
    1. Five logic inputs (A–E)
    2. common clock input (K)
    3. asynchronous direct-reset input (RD)
    4. enable (EC)
- Using programmable MUXes connected to the SRAM programming cells, you can independently connect each of the two CLB outputs (X and Y) to the output of the flip-flops (QX and QY) or to the output of the combinational logic (F and G).

Xilinx CLB (cont.)

- The combinational function in a CLB is implemented with a 32 bit *look-up table* (LUT)
  - LUT values are stored in 32 bits of SRAM
  - CLB delay is fixed and equal to the LUT access time
- 32-bit LUT requires only five variables to form a unique address (32 = 2 5)
- There are seven inputs to the LUT, the five CLB inputs (A-E) and the flip-flop outputs (QX and QY) and two outputs (F,G)
- There are several ways to use the LUT:
  - You can use five of the seven possible inputs (A-E,QX,QY) with the entire LUT - the outputs (F,G) are identical
  - You can split the 32-bit LUT in half to implement two functions of four variables
    - The input variable can be chosen from A-E,QX,QY
    - Two of the inputs must come from A-E. one function output connects to F and other to G.
  - You can split the LUT in half and use one of the seven input variables to select between the F and G output - allows some functions of seven variables to be implemented.

Xilinx XC4000 Logic Block

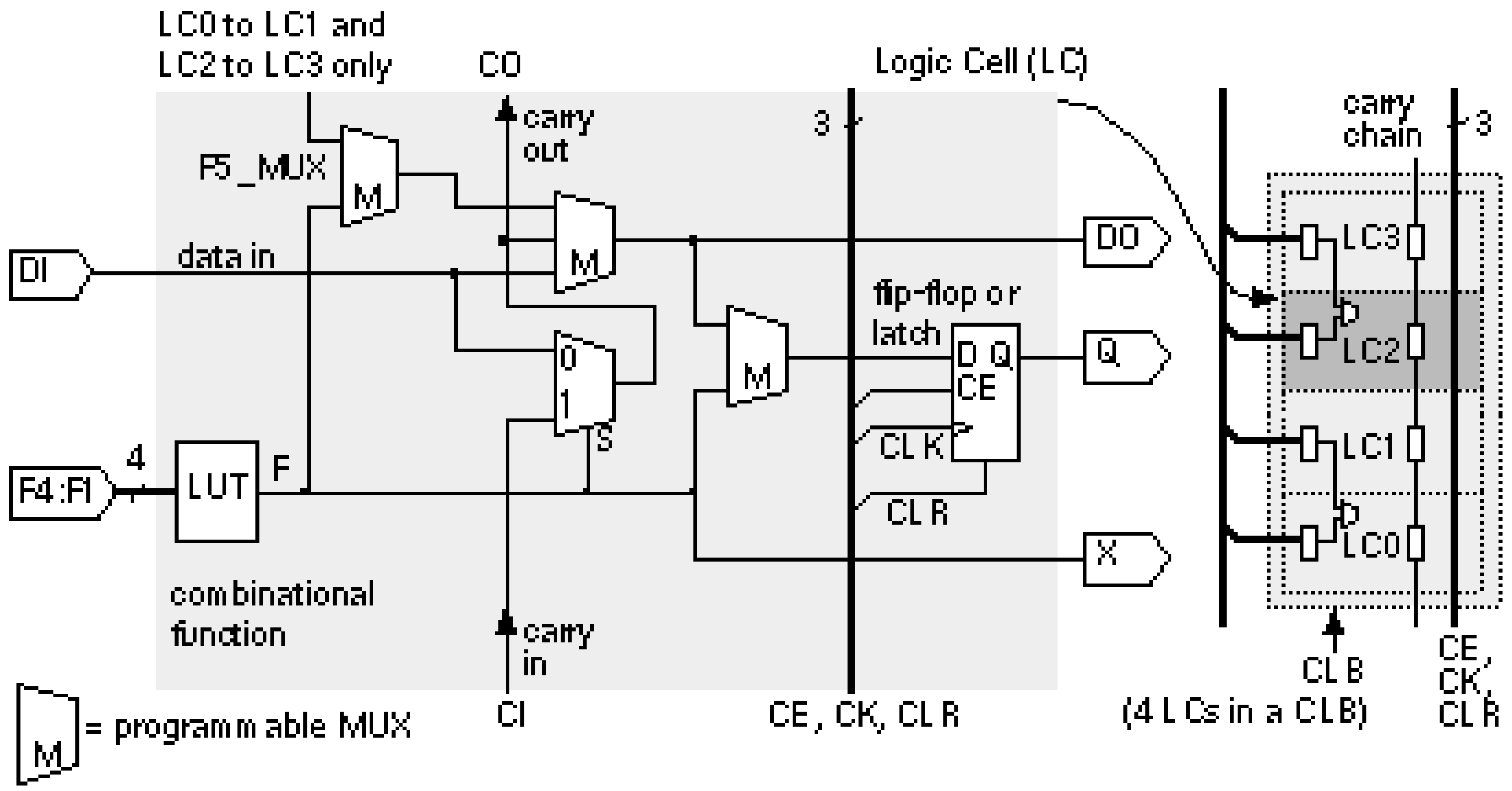


The Xilinx XC4000 CLB (configurable logic block).

- complicated basic logic cell containing **2 four-input LUTs** that feed **a three-input LUT.**
- The XC4000 CLB also has **special fast carry logic hard-wired** between CLBs.
- MUX control logic maps **four control inputs (C1-C4)** into the **four inputs:** LUT input **H1**, direct in (**DIN**), enable clock (**EC**), and a set / reset control (**S/R**) for the flip-flops.
- The control inputs (**C1-C4**) can also be used to control the use of the **F' and G'** LUTs as 32 bits of SRAM.

Xilinx XC5200 Logic Block

- Basic Cell is called a ***Logic Cell*** **(LC)** and is similar to, but simpler than, CLBs in other Xilinx families
- Term **CLB is** used here to mean **a group of 4 LCs** (LC0-LC3)

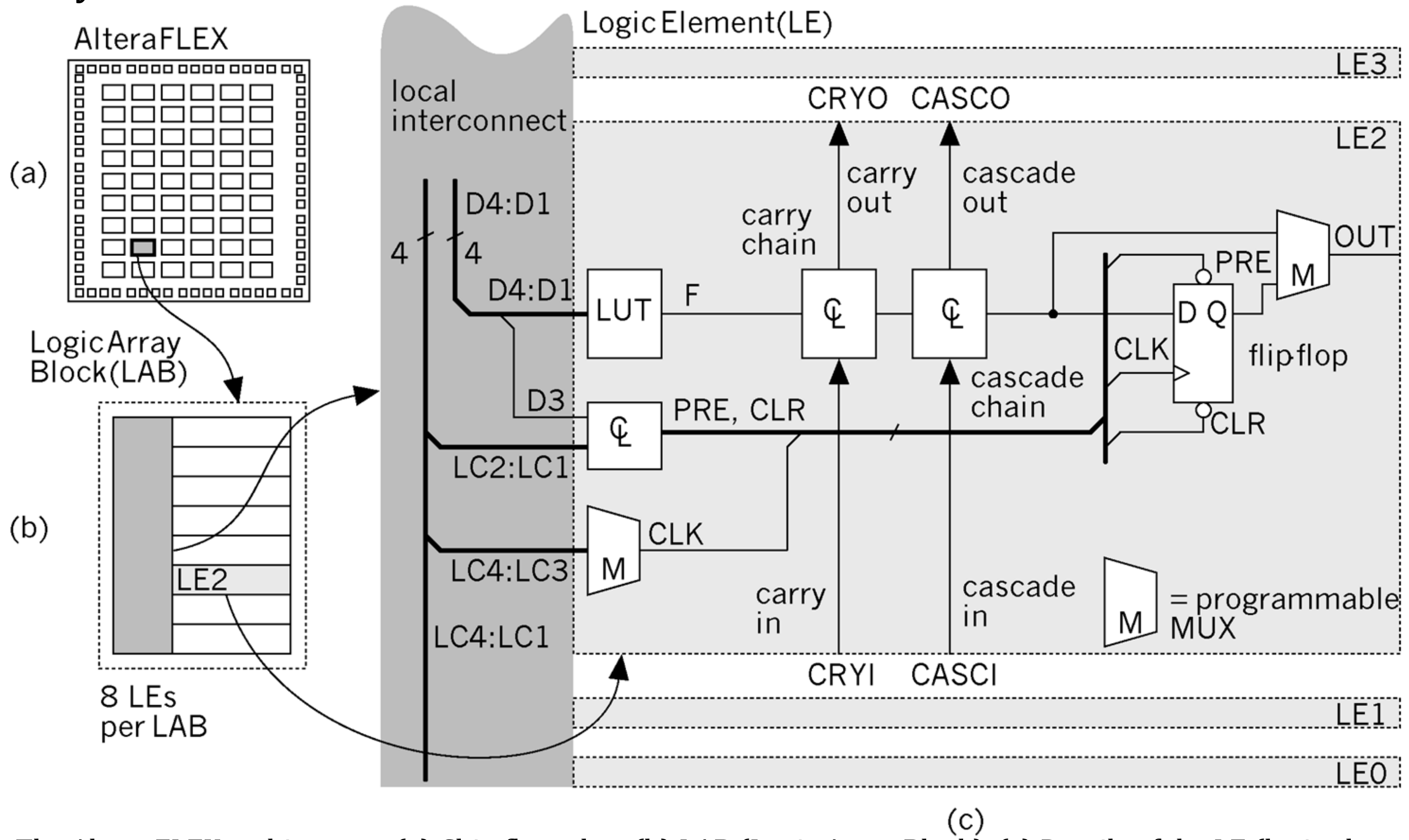


The Xilinx XC5200 LC (logic cell) and CLB (configurable logic block).

- The XC5200 LC contains a **four-input LUT, a flip-flop, and MUXes** to handle signal switching.
- The arithmetic **carry logic is separate from the LUTs.**
- A limited capability to cascade functions is provided (using the MUX labeled F5_MUX in logic cells LC0 and LC2 in Fig) **to gang two LCs in parallel to provide the equivalent of a five-input LUT.**

Altera FLEX Architecture

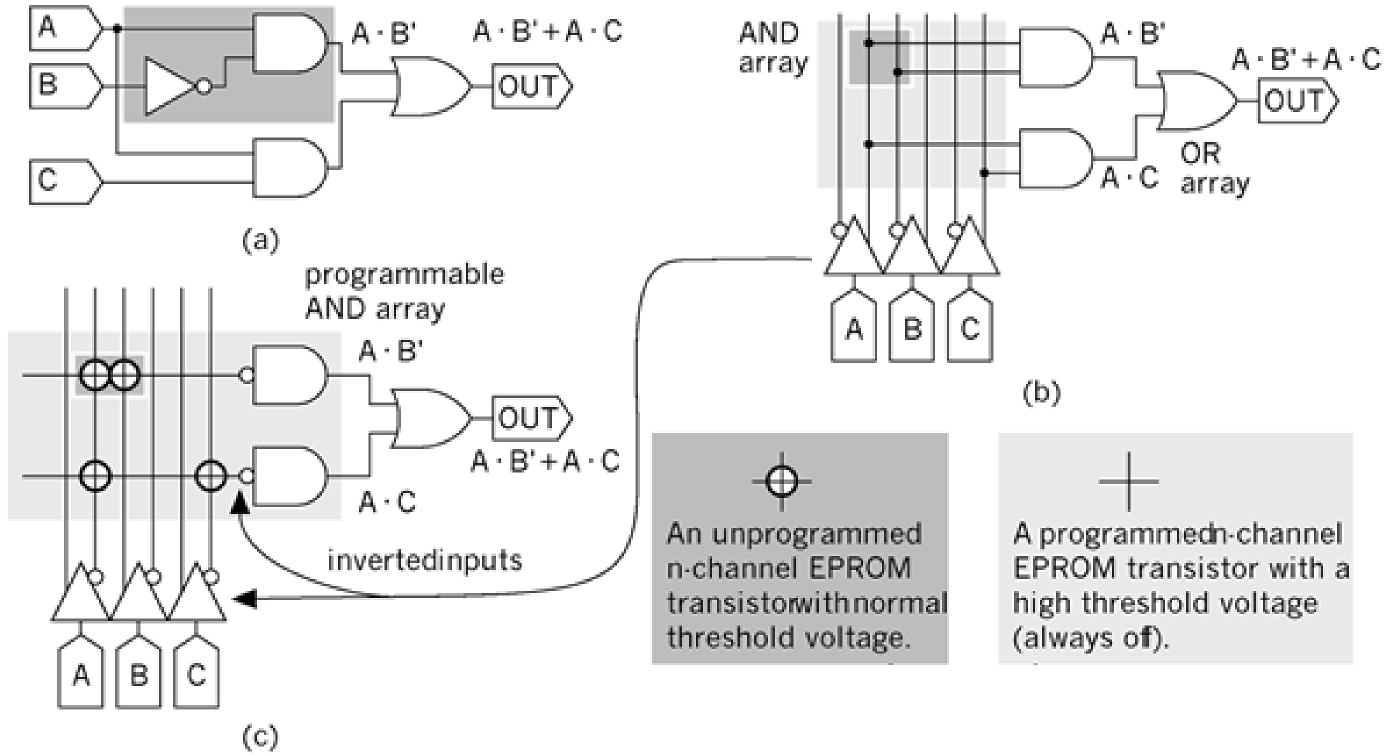
- Basic Cell is called a ***Logic Element* (LE)** that Altera uses in its FLEX 8000 series of FPGAs.
- FLEX resembles the Xilinx XC5200 LC architecture
- Altera FLEX uses the same SRAM programming technology as Xilinx
- The FLEX LE uses a four-input LUT, a flip-flop, cascade logic, and carry logic. Eight LEs are stacked to form a Logic Array Block



The Altera FLEX architecture. (a) Chip floorplan. (b) LAB (Logic Array Block). (c) Details of the LE (logic element).

Altera MAX

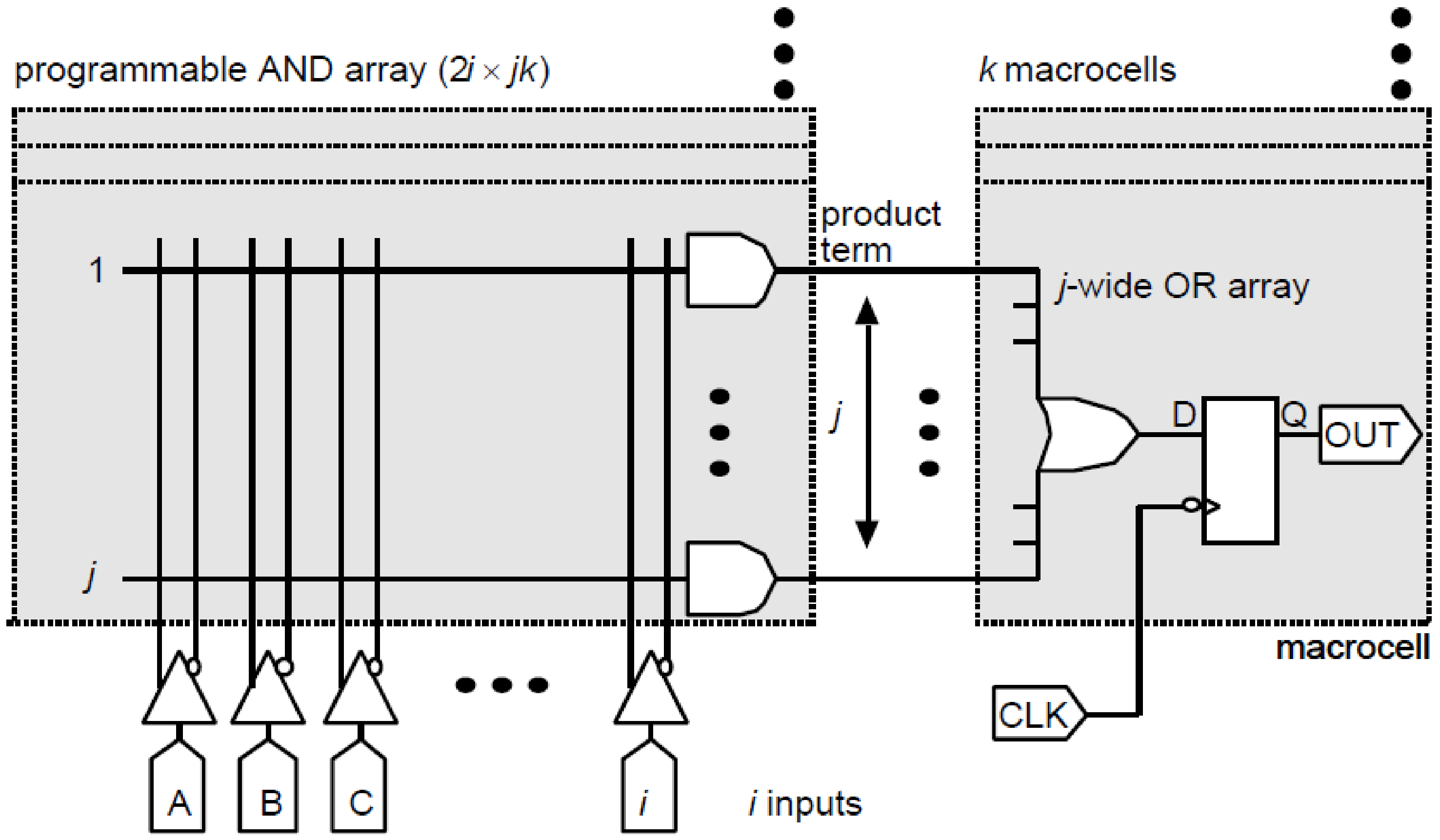
Programmable Logic Array


A
B
C
A · B'
A · B' + A · C
OUT
(a)
AND array
A · B'
A · B' + A · C
OUT
OR array
A · C
A
B
C
(b)
programmable AND array
A · B'
OUT
A · B' + A · C
A · C
invertedinputs
A
B
C
(c)
An unprogrammed n-channel EPROM transistorwithnormal threshold voltage.
A programmedn-channel EPROM transistor with a high threshold voltage (always off).

Logic Arrays. (a) Two-level logic. (b) Organized sum of products. (c) A programmable-AND plane

Figure:
a: simple two-level logic circuit that implements a sum of products.
b: a vector of buffers, followed by a vector of AND gates (which construct the product terms) that feed OR gates (which form the sums of the product terms).
c: the input lines to a multiple-input AND gate as if they were one horizontal wire, which we call a product-term line - called programmable array logic

(c) is very similar to a ROM, we sometimes call a horizontal product-term line, which would be the bit output from a ROM, the **bit line** . The vertical input line is the **word line** .



A registered PAL with *i* inputs, *j* product terms, and *k* macrocells. (*Source:* Altera (adapted with permission).)

Logic Expander

- A ***logic expander*** is an output line of the AND array that feeds back as an input to the array itself
- Logic expanders can help implement functions that require more product terms.
- Logic expanders and expander terms (helper terms) increase term efficiency
- Consider implementing this function in a three-wide OR array:

$$F = A' \cdot C \cdot D + B' \cdot C \cdot D + A \cdot B + B \cdot C'$$

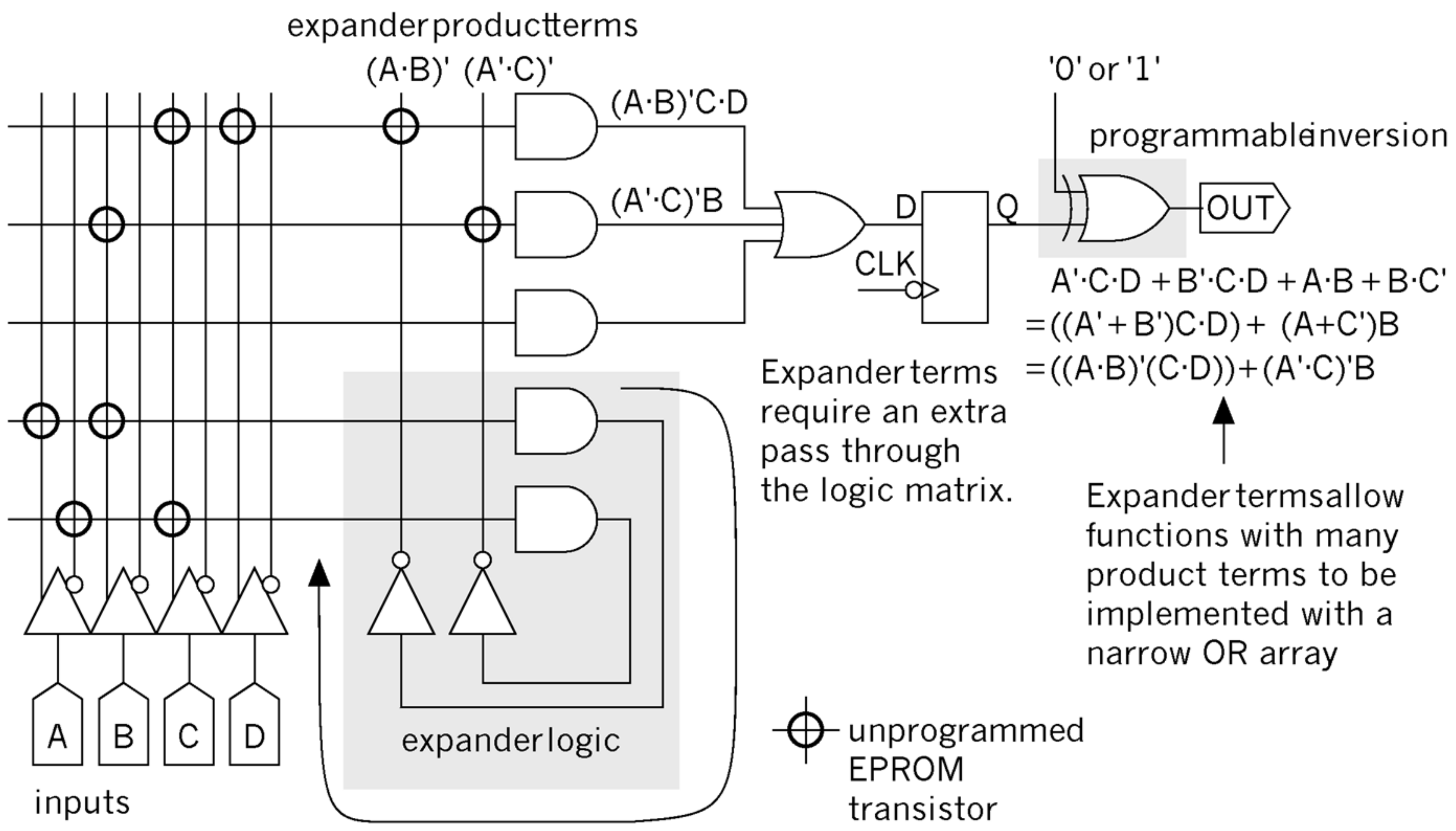
- This can be rewritten as a "sum of products :

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

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

- Logic expanders can be used to form the expander terms $(A \cdot B)'$ and $(A' \cdot C)'$
- Logic expanders require an extra pass through the AND array, increasing delay

Logic Expander Implementation

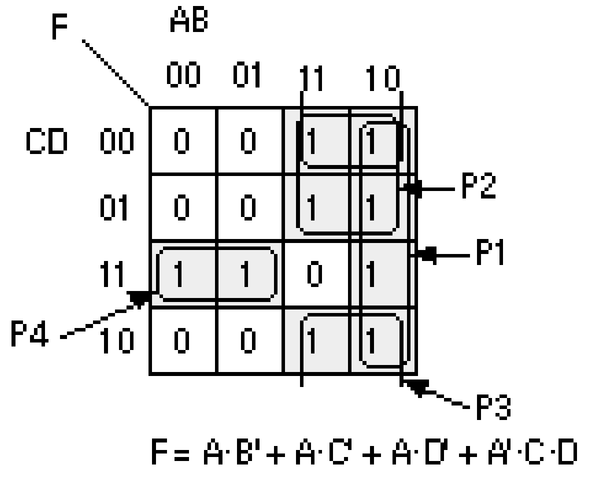

expander product terms
(A·B)' (A'·C)'
(A·B)'C·D
(A'·C)'B
'0' or '1'
programmable inversion
D
Q
CLK
OUT
A'·C·D + B'·C·D + A·B + B·C'
=((A' + B')C·D) + (A+C')B
=((A·B)'(C·D))+(A'·C)'B
Expander terms require an extra pass through the logic matrix.
Expander terms allow functions with many product terms to be implemented with a narrow OR array
A
B
C
D
inputs
expander logic
unprogrammed EPROM transistor

Expander logic and programmable inversion.

- We can even share these extra product terms with other macrocells if we need to.
- We call the extra logic gates that form these **shareable product terms** a **shared logic expander** , or just **shared expander** .

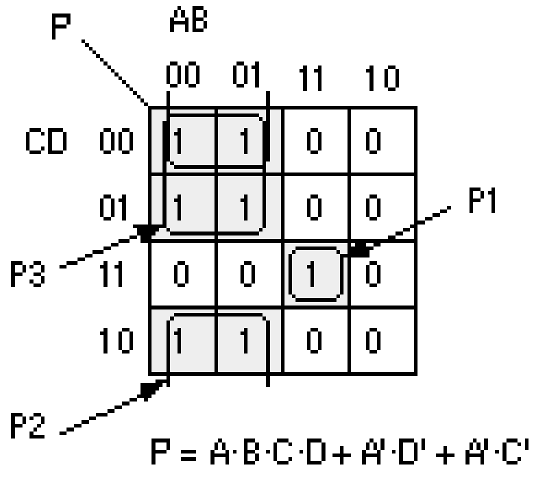
- The disadvantage of the shared expanders is the **extra logic delay** incurred because of the second pass that you need to take through the product-term array.

Programming one input of the XOR gate at the macrocell output allows you to choose whether or not to invert the output (a '1' for inversion or to a '0' for no inversion). This programmable inversion can reduce the required number of product terms



CD \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	0	1
10	0	0	1	1

(a)

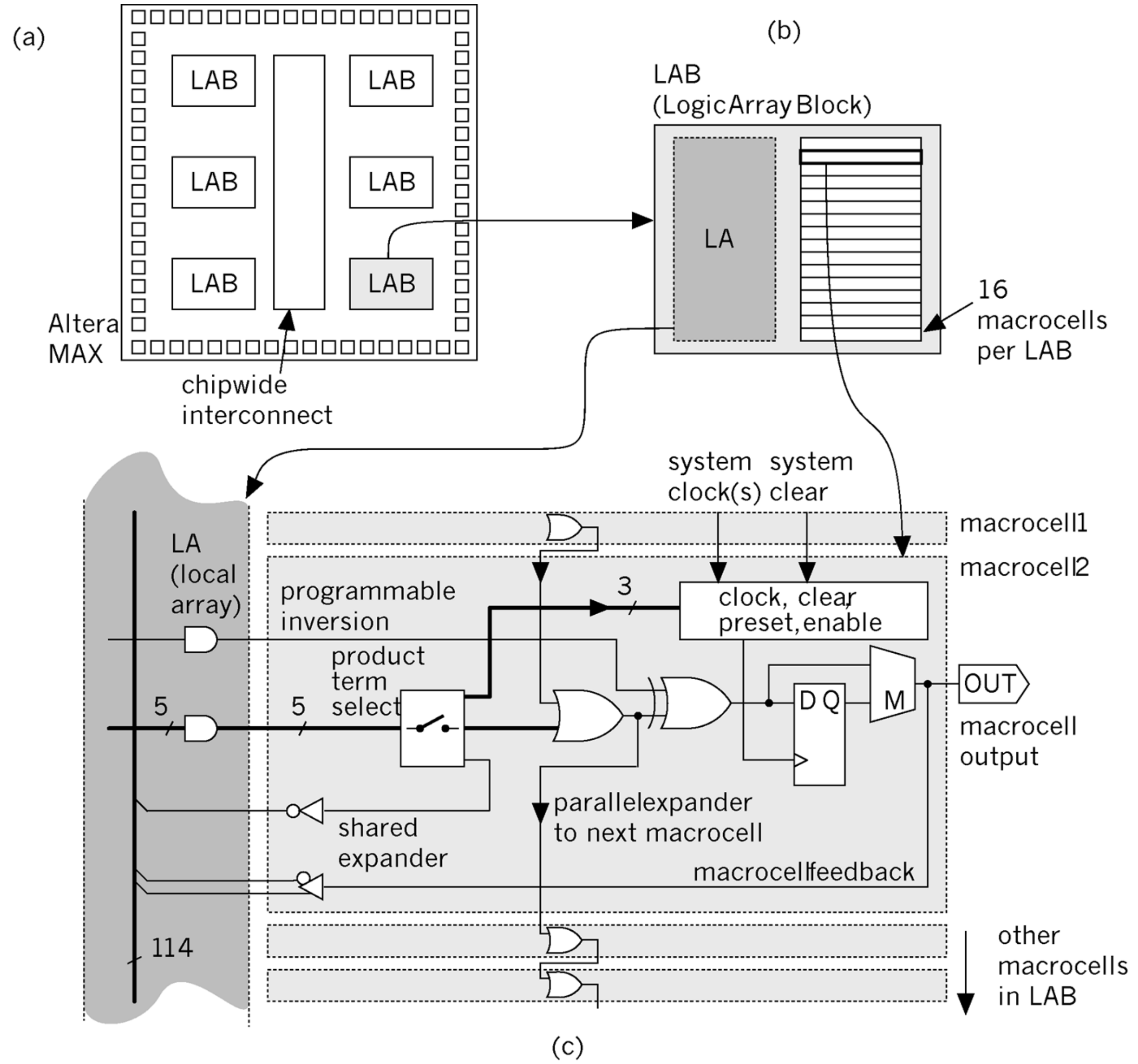


CD \ AB	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	0	1	0
10	1	1	0	0

(b)

- F requires four product terms—one too many for a three-wide OR array.
- F ' has only three product terms.
- To create F we invert F ', using programmable inversion.

Altera MAX Architecture



The Altera MAX architecture. (a) Organization of logic and interconnect. (b) A MAX family LAB (Logic Array Block). (c) A MAX family macrocell.

- Altera MAX macrocell and illustrates the architectures of several different product families.
- The implementation details vary among the families, but the basic features:
  - Wide, programmable AND array
  - Narrow, fixed OR array
  - Logic Expanders
  - Programmable inversion

Programmable ASIC I/O cells: DC output, AC output, DC input, AC input, Clock input, Power input, Xilinx I/O Block, other I/O cells. (6.1,6.2,6.3,6.4,6.5,6.6,6.7 & 6.8)

I/O Requirements

- I/O cells handle driving signals off chip, Receiving and conditioning external inputs, Supplying power and ground.
- Handling such things as electrostatic protection
- Different types of I/O requirements
  - *DC output* - **driving a resistive load** at DC or low frequency(less than 1MHz). Ex: LEDs, relays, small motors, etc..
  - *AC output* - **driving a capacitive load** with a high-speed (greater than 1MHz) logic signal off-chip, data or address bus, serial data line, etc.
  - *DC input* - **reading the value of a sensor, switch, or another logic chip**
  - *AC input* - **reading the value of high-speed signals** from another chip
  - *Clock input* - **system or synchronous bus inputs**
  - *Power input* - **supplying power** (and ground) to the I/O cells and logic core

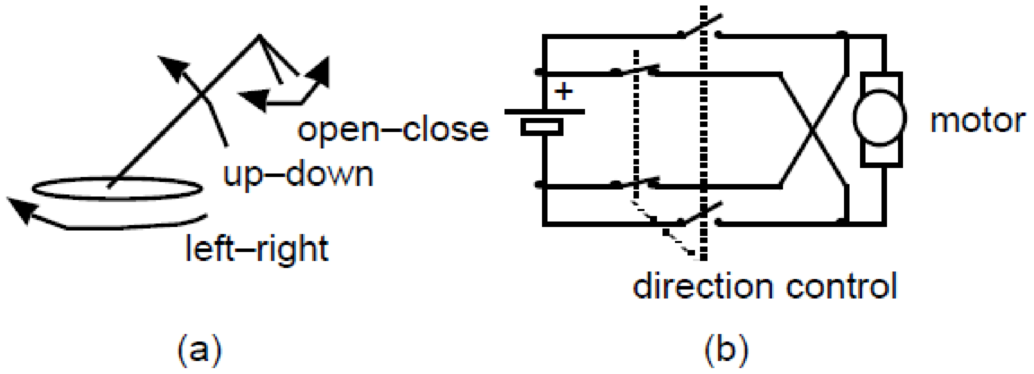
DC output

A robot arm example

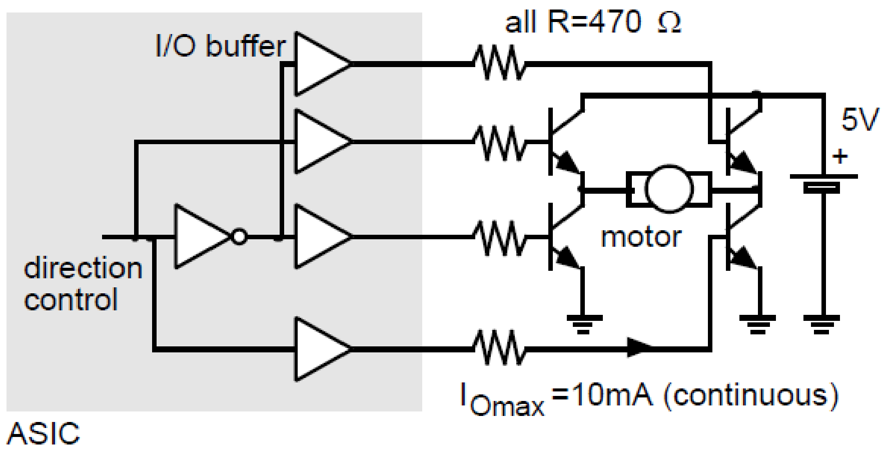
To design a system work from the outputs back to the inputs

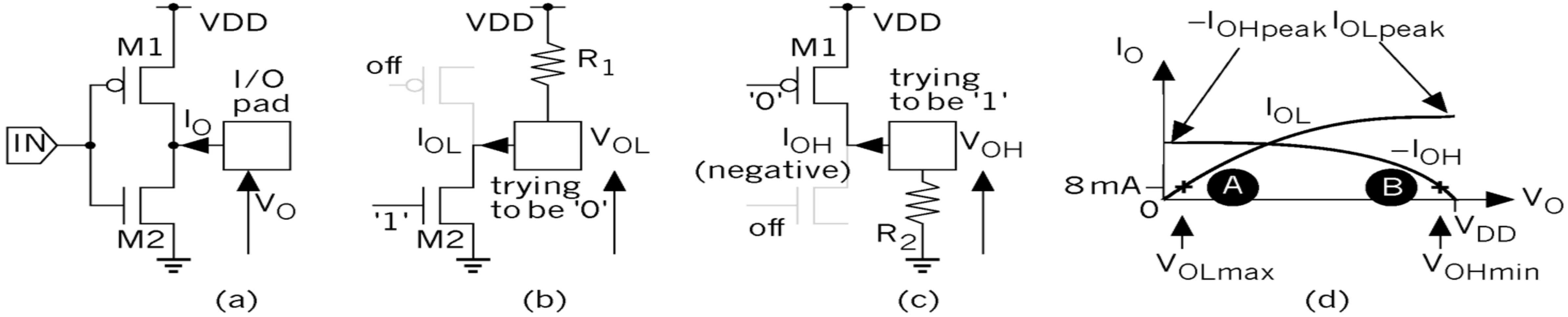
(a) Three small DC motors drive the arm

(b) Switches control each motor



- A circuit to drive a small electric motor (0.5A) using ASIC I/O buffers
- Work from the outputs to the inputs.
- The 470W resistors drop up to 5V if an output buffer current approaches 10mA, reducing the drive to the output transistors





CMOS output buffer characteristics:

(a) A CMOS complementary output buffer

(b) Transistor M2 (M1 off) sinks (to GND) a current IOL through a pull-up resistor, R1

(c) Transistor M1 (M2 off) sources (from VDD) a current –IOH (IOH is negative) through a pull-down resistor, R2

(d) Output characteristics:

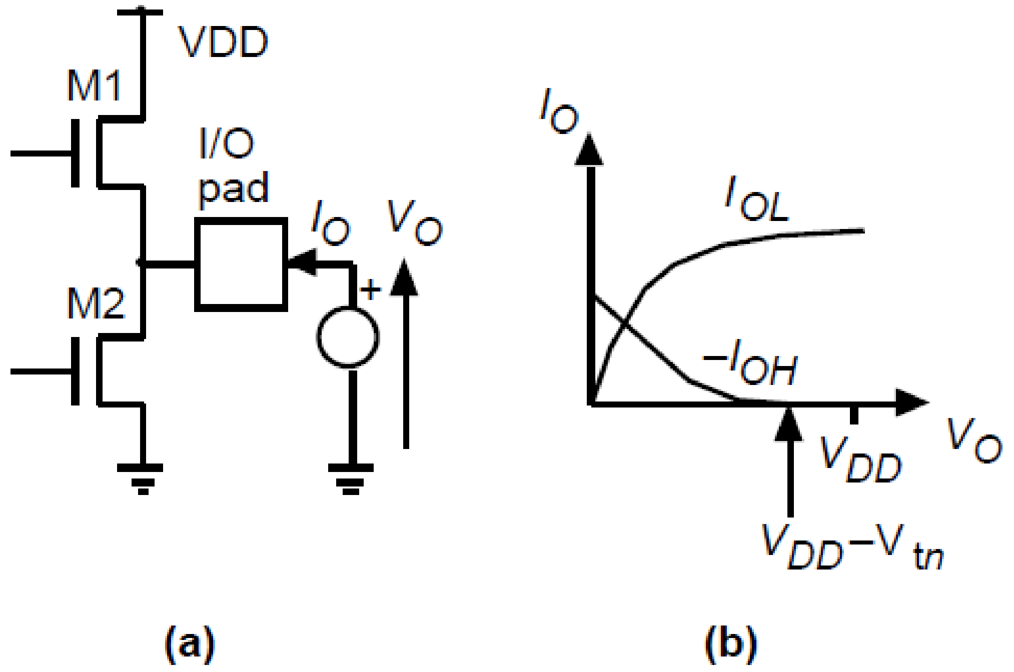
Data books specify characteristics at two points, A (VOHmin, IOHmax) and B (VOLmax,IOLmax)

Example (Xilinx XC5200):

VOLmax=0.4V, low-level output voltage at IOLmax=8.0mA

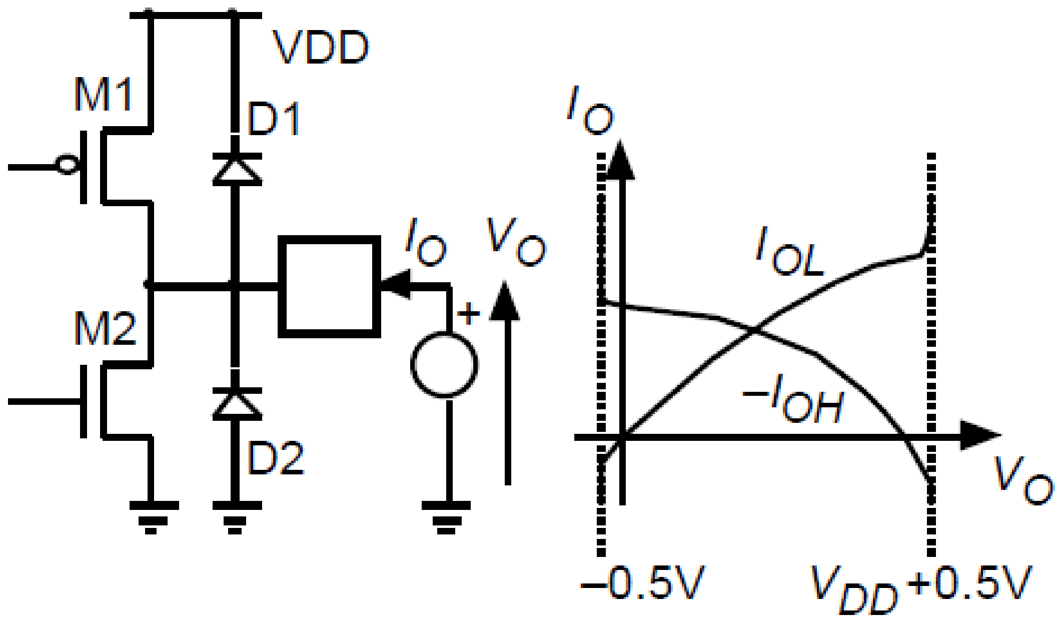
VOHmin=4.0V, high-level output voltage at IOHmax=–8.0mA

- **Totem –Pole Output buffer:** has two stacked transistors of the same type and the complementary output uses transistor as opposite types.
- The high-level voltage, VOHmin , for a totem pole is lower than VDD . Typically VOHmin is in the range of 3.5 V to 4.0 V (with VDD = 5 V), which makes rising and falling delays more symmetrical and more closely matches TTL voltage levels.
- The disadvantage is that the totem pole will typically only drive the output as high as 3–4 V; so this would not be a good choice of FPGA output buffer to work



Output buffer characteristics
(a) A CMOS totem-pole output stage (both M1 and M2 are n-channel transistors)
(b) Totem-pole output characteristics (notice the reduced signal swing)

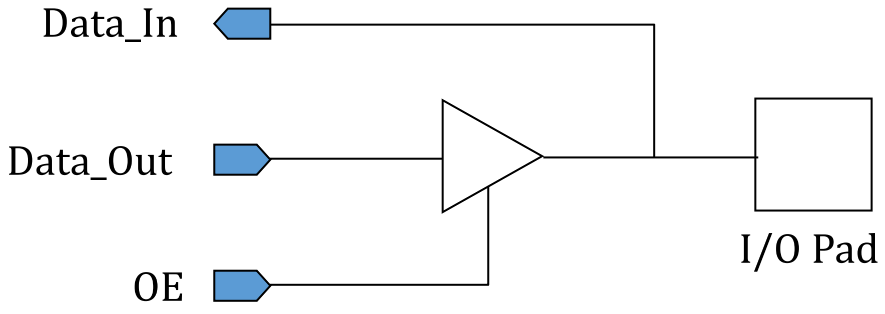
- **Clamp Diodes:** D1 and D2, in an output buffer (totem-pole or complementary) prevent the I/O pad from voltage excursions greater than VDD and less than VSS
- The clamp diodes conduct as the output voltage exceeds the supply voltage bounds

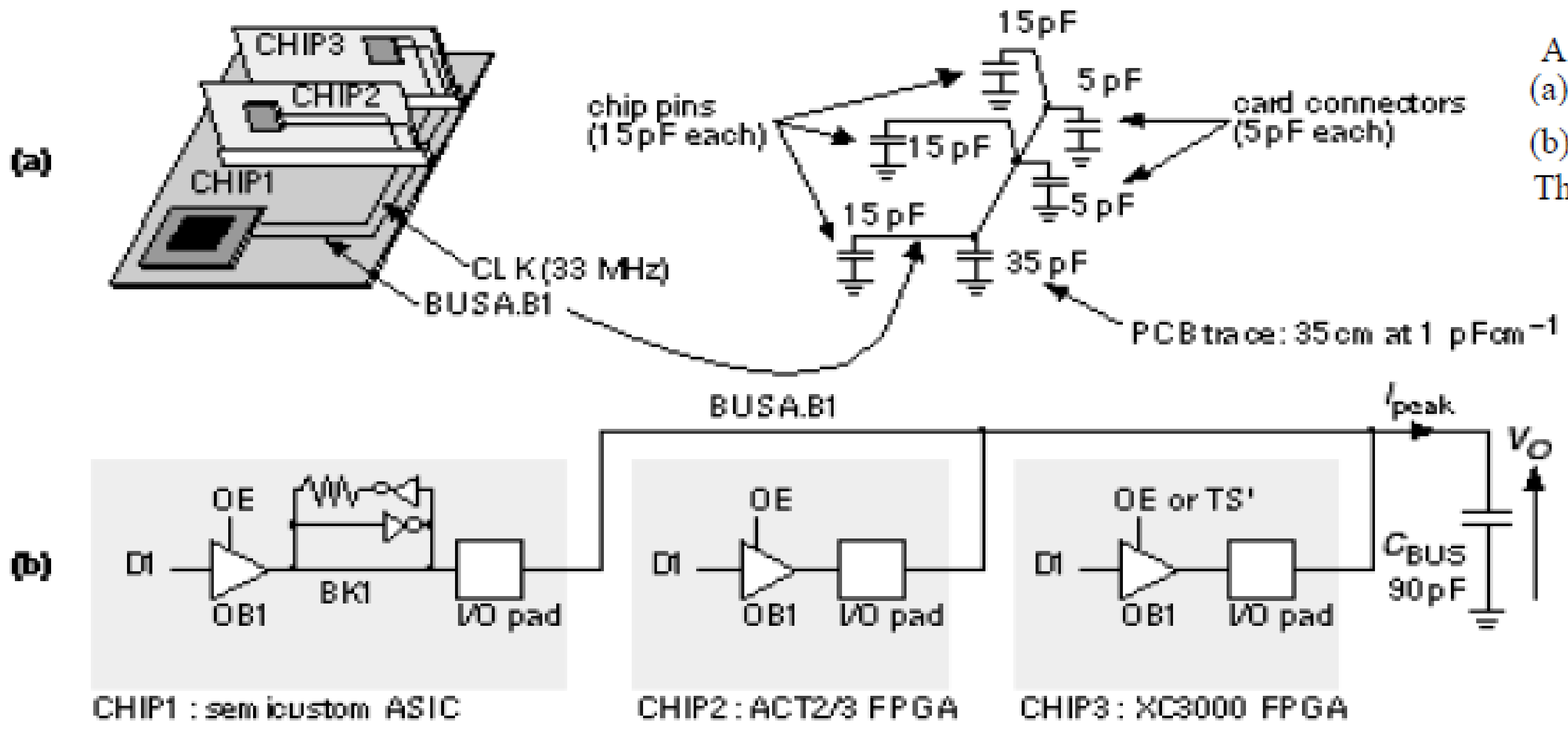


AC Output

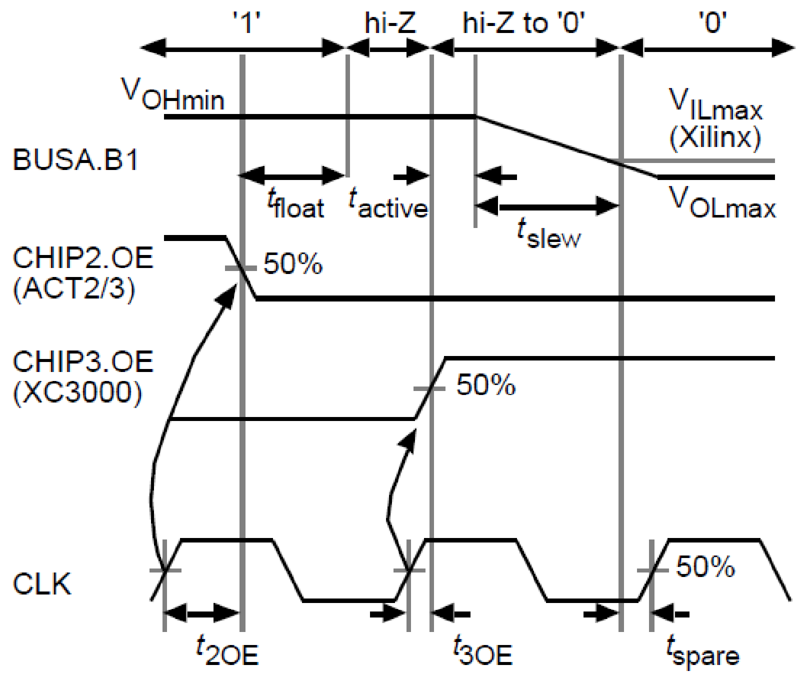
- Chips that have inputs and outputs connected to a bus are called **bus transceivers .**
- AC outputs are often used to connect **to a bi-directional bus** - bus transceivers
- This functionality requires the capability for *three-state* (tri-state) outputs - ‘0’, ‘1’, and *high-impedance* or *hi-z*
- In addition to rise and fall times, bidirectional I/O pads have timing parameters related to the hi-z state *(float time):*
  - t_{ENZL} - output hi-Z to ‘0’ time
  - t_{ENLZ} - output ‘0’ to hi-Z
  - t_{ENZH} - output hi-Z to ‘1’
  - t_{ENHZ} - output ‘1’ to hi-Z

Bi-Directional I/O Pad





A three-state bus.
(a) Bus parasitic capacitance.
(b) The output buffers in each chip.
The ASIC CHIP1 contains a bus keeper, BK1.



Three-state bus timing

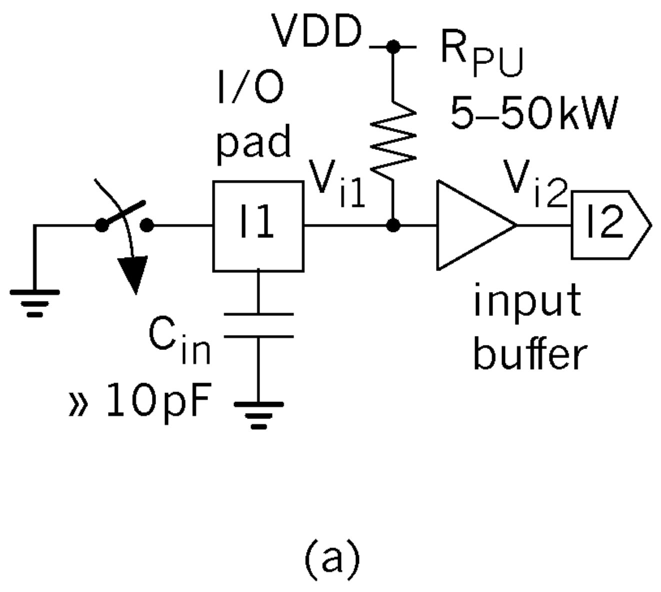
The on-chip delays, t $_{2OE}$ and t $_{3OE}$, for the logic that generates signals CHIP2.E1 and CHIP3.E1 are derived from the timing models

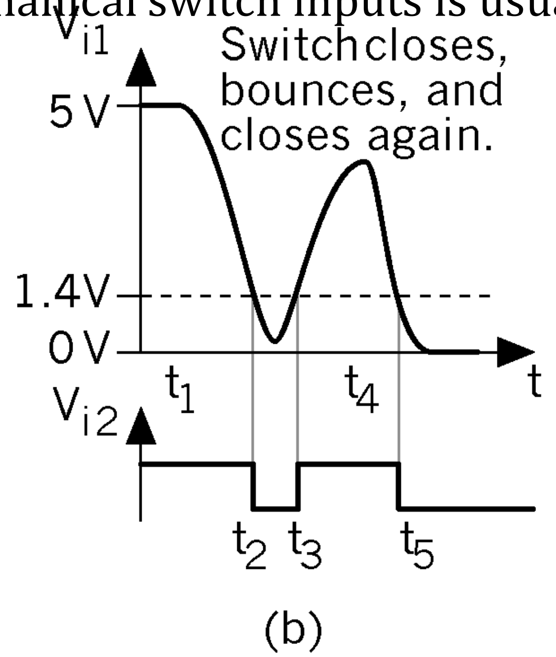
timing of part of a bus transaction

1. Initially CHIP2 drives BUSA.B1 high (CHIP2.D1 is '1' and CHIP2.OE is '1').
2. The buffer output enable on CHIP2 (CHIP2.OE) goes low, floating the bus. The bus will stay high because we have a bus keeper, BK1.
3. The buffer output enable on CHIP3 (CHIP3.OE) goes high and the buffer drives a low onto the bus (CHIP3.D1 is '0').

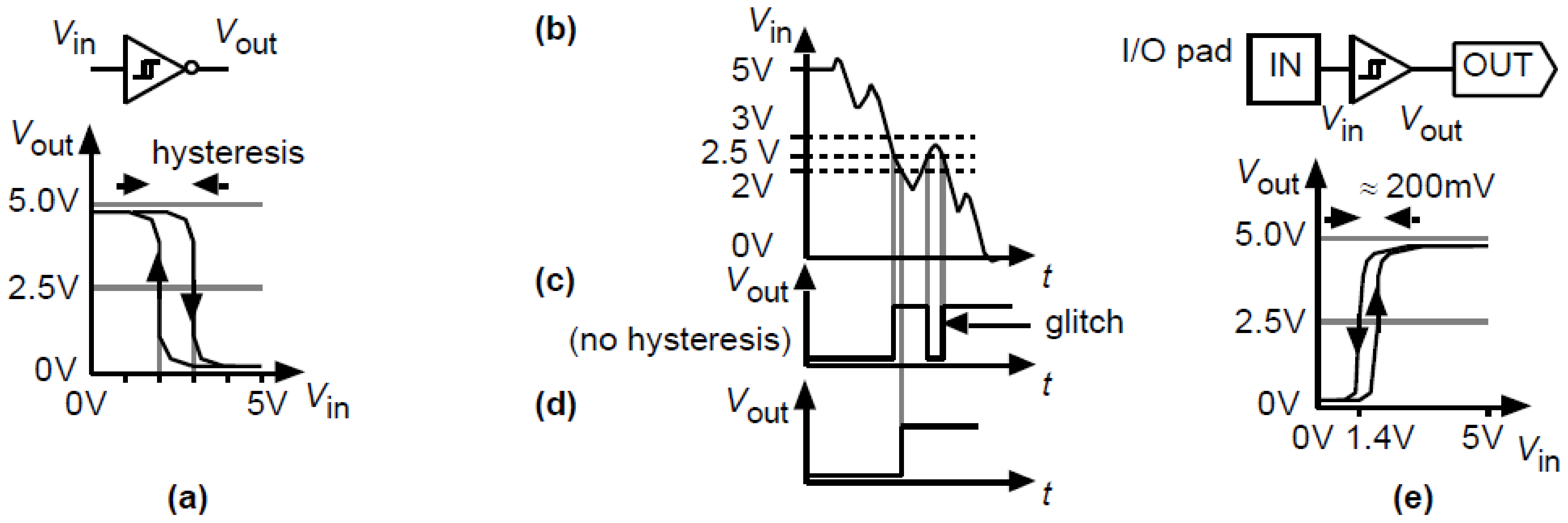
DC Input - Switch Bounce

- Suppose, a pushbutton switch is connected to the input of an FPGA. Most FPGA input pads are directly connected to a buffer.
- optional Pull-up registers – used to ensure the buffer never floats the voltage between the valid logic levels.
- A pull-up or pull-down resistor is generally required on input buffers to keep input from floating to indeterminate logic levels
- If the input is from a mechanical switch, the contacts may bounce, producing several transitions through the switching threshold. A bouncing switch may create a noisy waveform in the time domain, we may also have noise in the voltage level of our input signal.
- Some technique for *debouncing* mechanical switch inputs is usually necessary



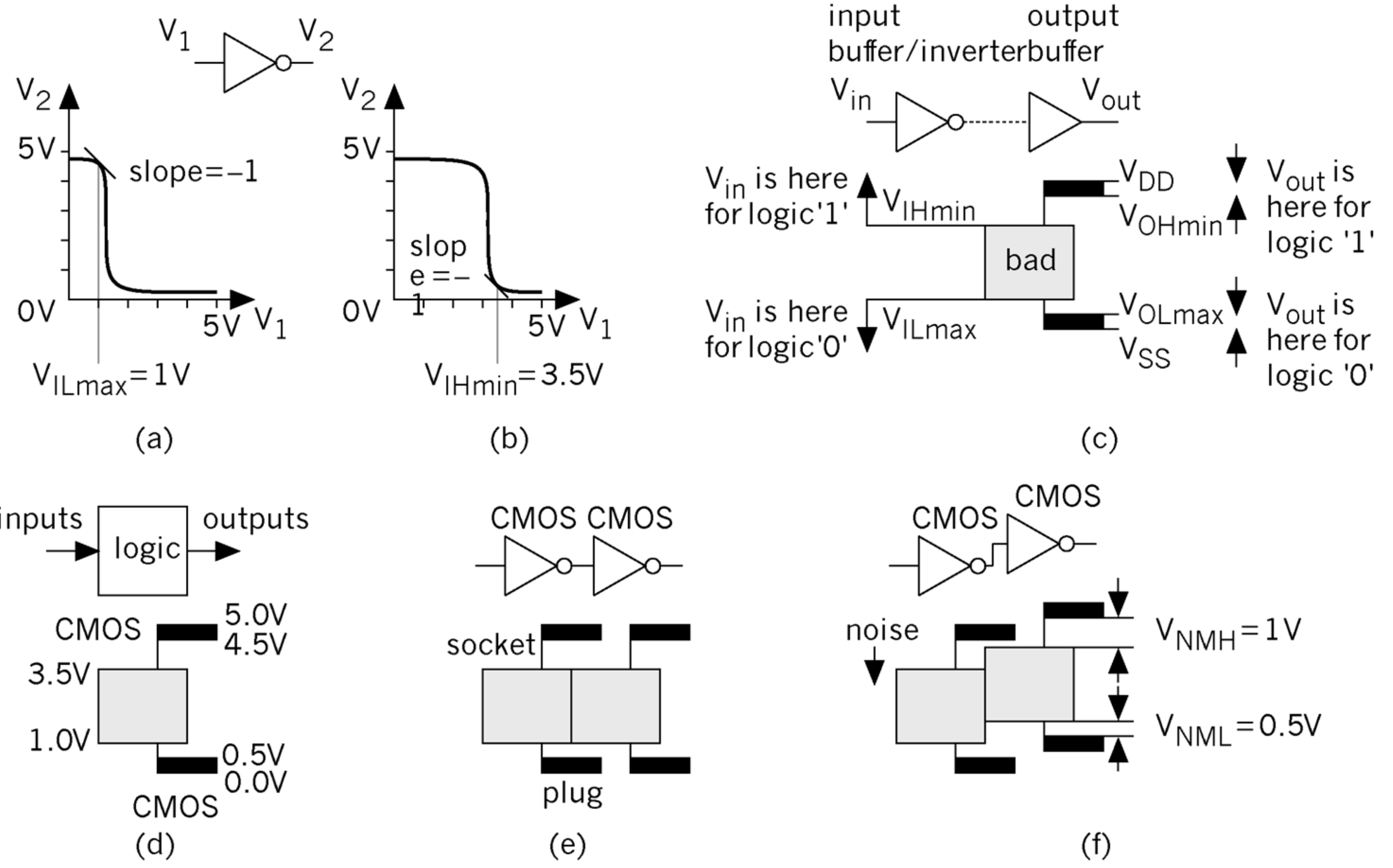


A switch input. (a) A pushbutton switch connected to an input buffer with a pull-up resistor. (b) As the switch bounces several pulses may be generated using SR flip flop..



- The Schmitt-trigger inverter in Figure (a) has a lower switching threshold of 2 V and an upper switching threshold of 3 V. The difference between these thresholds is the **hysteresis** , equal to 1 V in this case.
- If we apply the **noisy waveform** shown in Figure (b) to an inverter **with no hysteresis**, there will be a **glitch at the output**, as shown in Figure (c). As long as **the noise on the waveform does not exceed the hysteresis**, the Schmitt-trigger inverter will produce the **glitch-free output** of Figure (d).
- Most FPGA input buffers **have a small hysteresis** (the 200 mV that Xilinx uses is a typical figure) centered around 1.4 V, as shown in Figure (e).
- Hysteresis in the input buffer also **helps prevent oscillation and noise problems with inputs that have slow rise times.**

Noise Margins - Another Representation



Noise margins.

(a) Transfer characteristics of a CMOS inverter with the **lowest switching threshold.**

(b) The **highest switching threshold**

(c) A graphical representation **of CMOS thresholds.**

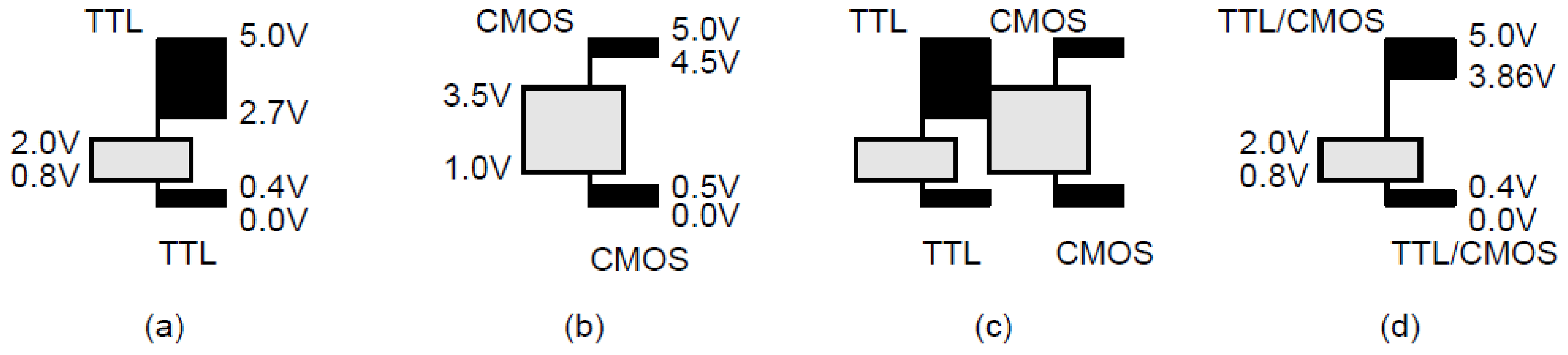
(d) Logic thresholds at the inputs and outputs of a logic gate or an ASIC.

(e) The switching thresholds viewed as a plug and socket. If a plug fits a socket, we can connect the two components together and they will have compatible logic levels.

(f) CMOS plugs fit CMOS sockets and the clearances are the **noise margins.**

(c) depicts the following relationships between the various voltage levels at the inputs and outputs of a logic gate:

- A logic '1' output must be between V_{OHmin} and V_{DD}.
- A logic '0' output must be between V_{SS} and V_{OLmax}.
- A logic '1' input must be above the high-level input voltage , V_{IHmin}.
- A logic '0' input must be below the low-level input voltage , V_{ILmax}.
- Clamp diodes prevent an input exceeding V_{DD} or going lower than V_{SS}.



TTL and CMOS logic thresholds

(a) TTL logic thresholds

(b) Typical CMOS logic thresholds

(c) A TTL plug will not fit in a CMOS socket

- The lowest permissible TTL output level, V OHmin = 2.7 V, is too low to be recognized as a logic '1' by the CMOS input.

(d) Raising VOHmin solves the problem

- This is fixed by most FPGA manufacturers by raising V OHmin to around 3.8–4.0 V.

AC Input

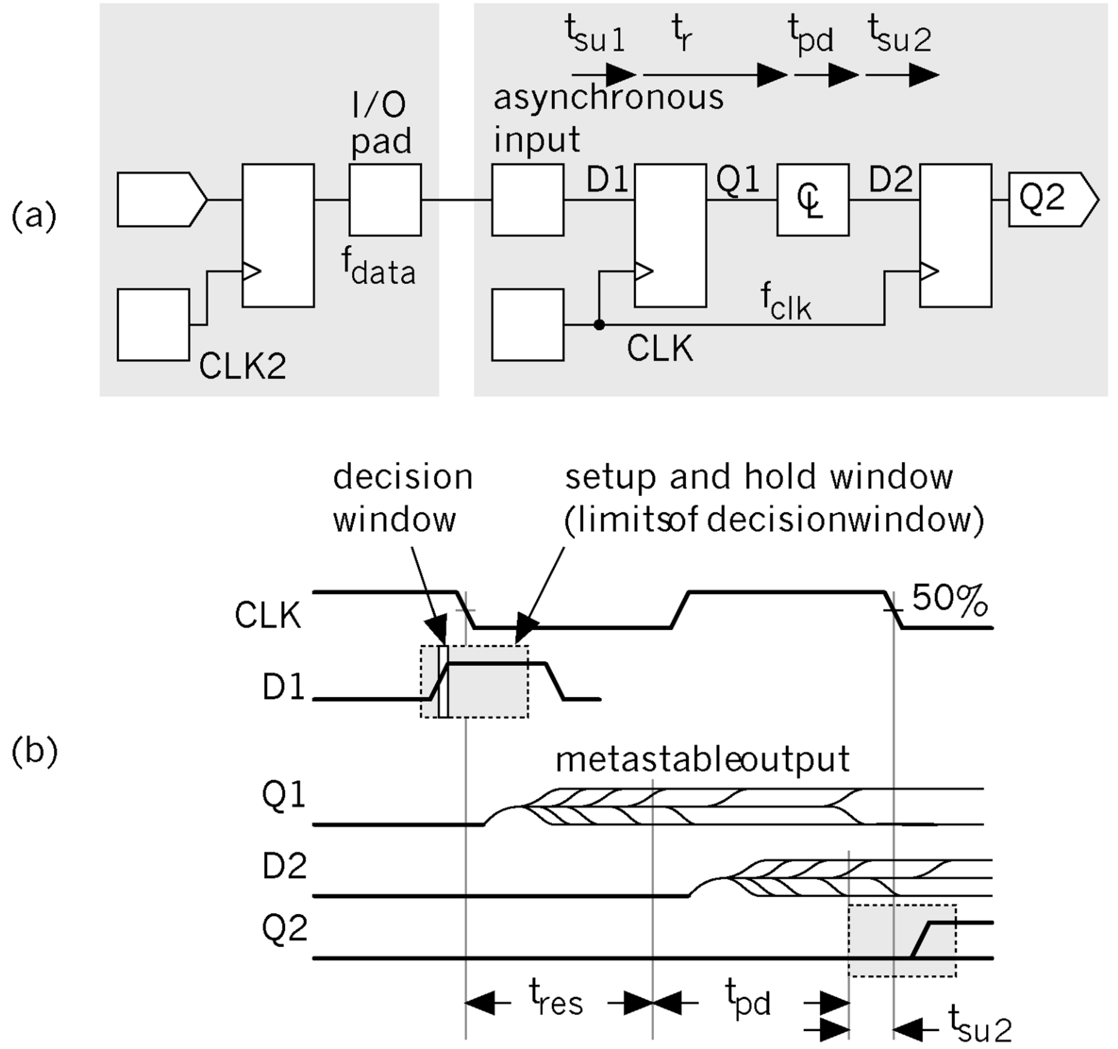
- Suppose we wish to **connect** an input bus containing **sampled data from an analog-to-digital converter (A/D)** that is running at a **clock frequency of 100 kHz** to an FPGA that is running from a **system clock on a bus at 10 MHz (NuBus).**

- We **cannot just connect** the A/D output bus to our FPGA, because we have **no idea when the A/D data will change**.

- We should have the **A/D data at the flip-flop input for at least the flip-flop setup time before the NuBus clock edge.**

- Unfortunately there is **no way to guarantee this**; the A/D converter clock and the NuBus clock are completely independent. Thus it is entirely possible that every now and again the **A/D data will change just before the NuBus clock edge**.

Metastability Example



- If we change the data input to a flip-flop (or a latch) too close to the clock edge(called a setup or hold-time violation), we run into a problem called **Metastability.**
- In this situation the flip-flop cannot decide whether its output should be a '1' or a '0' for a long time. If the flip-flop makes a decision, at a time tr after the clock edge, as to whether its output is a '1' or a '0', there is a small, but finite, probability that the flip-flop will decide the output is a '1' when it should have been a '0' or vice versa.
- This situation, **called an upset** , can happen when the data is coming from the outside world and the flip-flop can't determine when it will arrive; this is an asynchronous signal , because it is not synchronized to the chip clock.

Metastability. (a) Data coming from one system is an asynchronous input to another. (b) A flip-flop has a very narrow decision window bounded by the setup and hold times. If the data input changes inside this decision window, the output may be metastable - neither '1' or '0'.

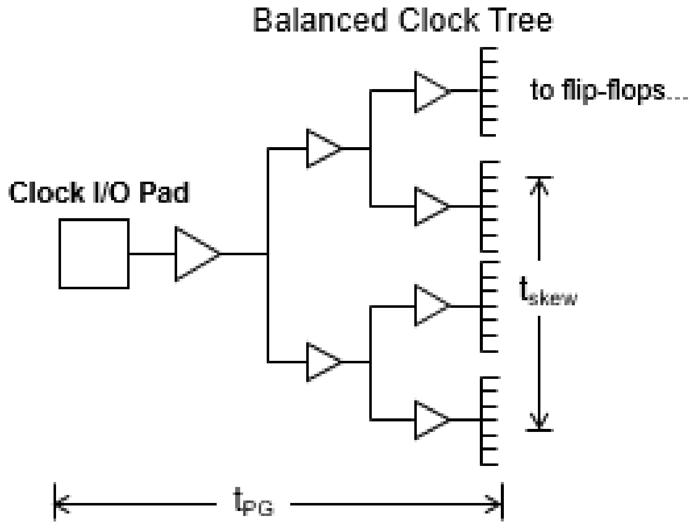
TABLE :Metastability parameters for FPGA flip-flops. These figures are not guaranteed by the vendors.

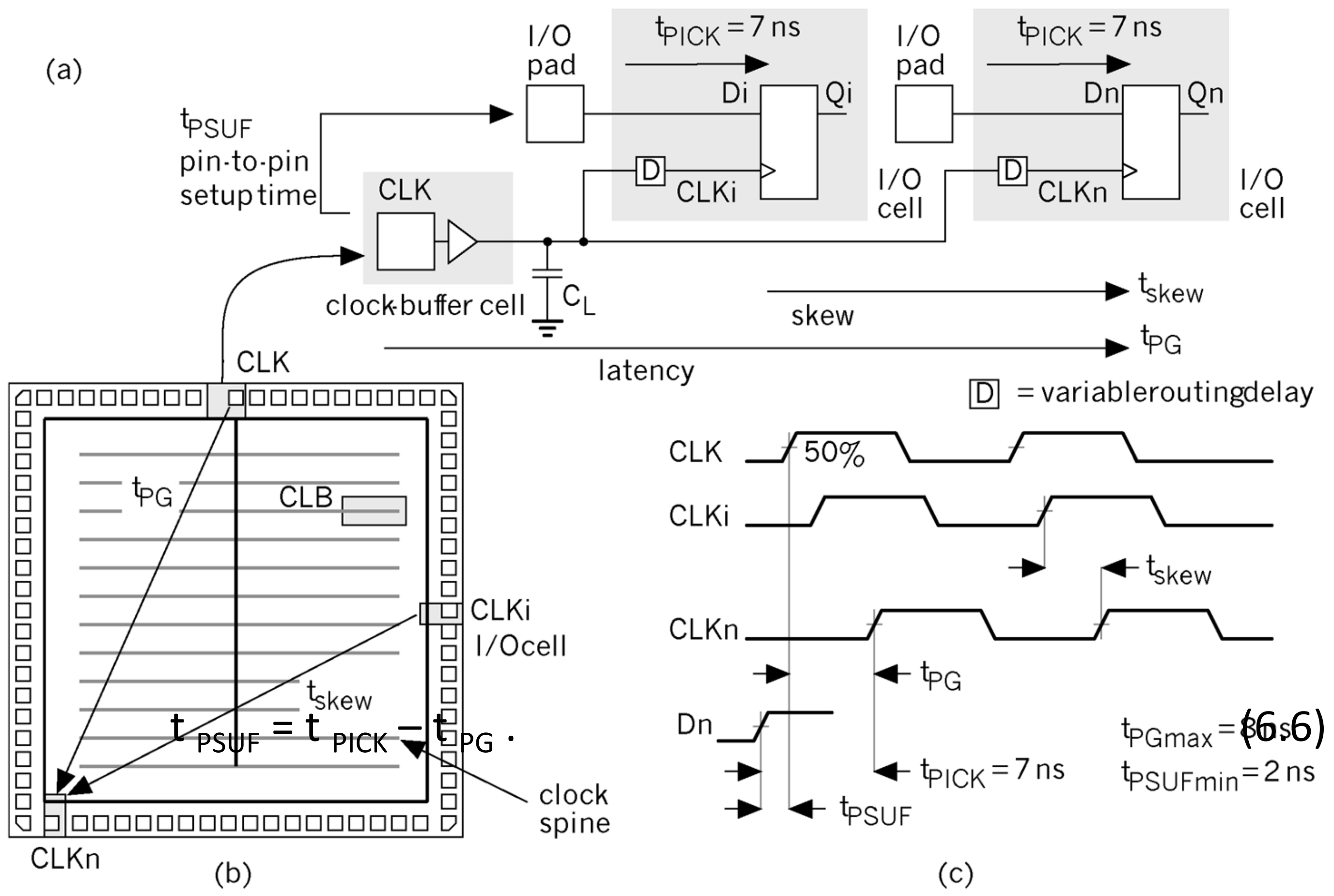
FPGA	T_0 / s	t_c / s
Actel ACT 1	1.0E09	2.17E10
Xilinx XC3020-70	1.5E10	2.71E10
QuickLogic QL12x16-0	2.94E11	2.91E10
QuickLogic QL12x16-1	8.38E11	2.09E10
QuickLogic QL12x16-2	1.23E10	1.85E10
Xilinx XC8100	2.15E-12	4.65E10
Xilinx XC8100 synchronizer	1.59E-17	2.07E10
Altera MAX 7000	2.98E17	2.00E10
Altera FLEX 8000	1.01E13	7.89E11

- The Parameter T_0 (unit of time) is a function of process technology and the circuit design
- The parameter t_c is the inverse of the gain bandwidth product, GB, of the sampler at the instance of sampling.

Clock Input

- When we bring the clock signal onto a chip, we may need to adjust the logic level and then we need to distribute the clock signal around the chip as it is needed.
- Most FPGAs and PLDs provide a **dedicated clock input(s).**
- We need to **minimize the clock delay** (or latency), but we also need **to minimize the clock skew.**
- Low skew is ensured by using a dedicated, balanced clock tree, but this tends to increase clock latency.
- Large clock latency causes hold time restrictions on data inputs - data gets to the flip-flops faster than clock and must remain there until clock arrives





$t_{PSUF} = t_{PICK} - t_{PG}$. (6.6)

Clock Input Example

(a) Timing model with values for Xilinx XC4005-6.

(b) A simplified view of clock distribution.

(c) Timing diagram. Xilinx eliminates the variable internal delay t_{PG} by specifying a pin-to-pin setup time $t_{PSUFmin}$ = 2ns.

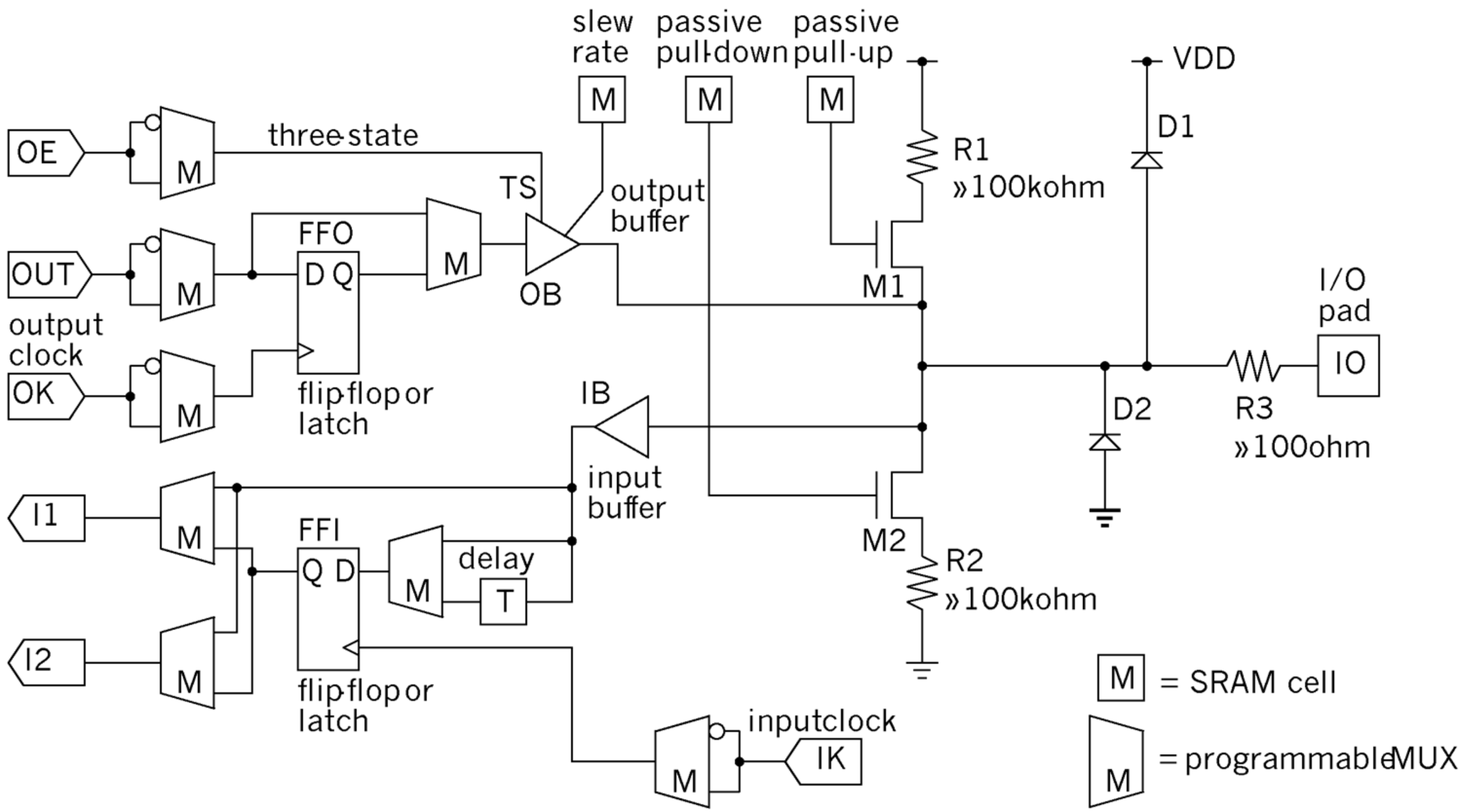
- t_{PICK} is the fixed setup time for a flip-flop relative to the flip-flop clock.
- t_{skew} is the variable clock skew , the signed delay between two clock edges.
- t_{PG} is the variable clock delay or latency .

To calculate the flip-flop setup time ($t_{PSUFmin}$) relative to the clock pad: $t_{PSUF} = t_{PICK} - t_{PG}$

Power Input

- All devices require inputs for **VDD and Gnd during operation** and **programming voltage, VPP, during programming**
- Larger devices with **greater logic capacity require more power pins** to supply the necessary power while maintaining a reasonable per-pin current limit
    - This reduces the number of signal pins possible for larger devices
- Some types of FPGAs (e.g. Xilinx) have their **own power-on reset** sequence to reset flip-flops, initialize and load SRAM, etc.
- **Power on reset:** FPGA configures all f.f’s as either SET or RESET. After chip programming is complete, the global SET/RESET forces all f.f’s on the chip to a known state.   - this may determine the initial state of a system

Example FPGA I/O Block



Xilinx XC4000 family IOB

The outputs contain features that allow you to do the following:

- Switch between a totem-pole and a complementary output (XC4000H).
- Include a passive pull-up or pull-down (both n -channel devices) with a typical resistance of about 50 k W .
- Invert the three-state control (output enable OE or three-state, TS).
- Include a flip-flop, or latch, or a direct connection in the output path.
- Control the slew rate of the output.

The features on the inputs allow you to do the following:

- Configure the input buffer with TTL or CMOS thresholds.
- Include a flip-flop, or latch, or a direct connection in the input path.
- Switch in a delay to eliminate an input hold time.