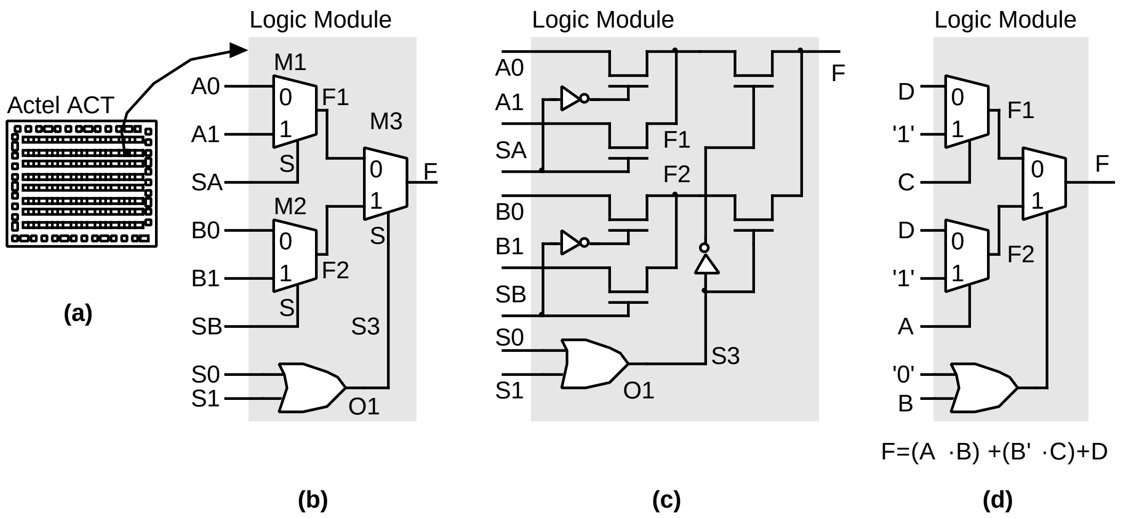
PROGRAMMABLE ASIC LOGIC CELLS

5

Key concepts: basic logic cell • multiplexer-based cell • look-up table (LUT) • programmable array logic (PAL) • influence of programming technology • timing • worst-case design

5.1 Actel ACT

5.1.1 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

5.1.2 Shannon’s Expansion Theorem

- We can use the **Shannon expansion theorem** to **expand** F =A·F(A='1') + A'·F(A='0')

Example: 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 F_A
- 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
- (A **minterm** is a **product term** that contains all the variables of F—such as A·B'·C)

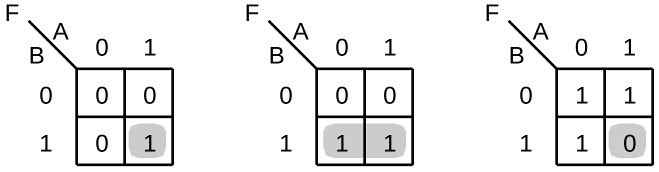
Another example: F=(A·B) + (B'·C) + D

- Expand F *wrt* B: F=B·(A + D) + B'·(C + D) =B·F2 + B'·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·1) + (A'·D); F1=C + D=(C·1) + (C'·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

5.1.3 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



4 ways to arrange one '1'

6 ways to arrange two '1's

4 ways to arrange one '0'

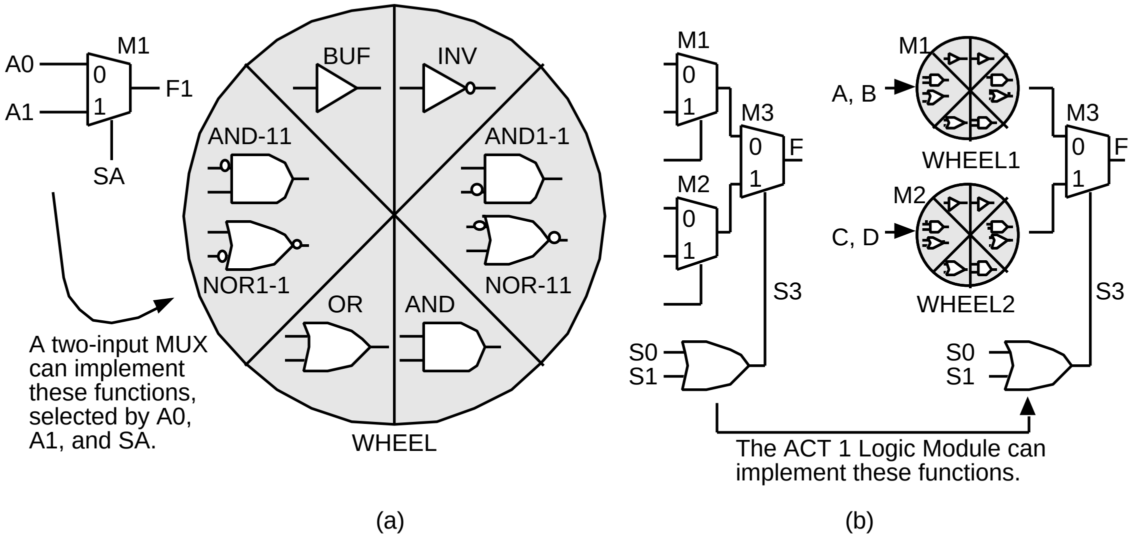
14 functions of 2 variables (and F='0', F ='1' makes 16)

Boolean functions using a 2:1 MUX

	Function, F	F=	Canonical form	Min-terms	Min-term code	Func-tion number	M1		
							A0	**A1**	**SA**
1	'0'	'0'	'0'	none	0000	0	0	0	0
2	NOR1-1(A, B)	(A+B')	A'·B	1	0010	2	B	0	A
3	NOT(A)	A'	A'·B' + A'·B	0, 1	0011	3	0	1	A
4	AND1-1(A, B)	A·B'	A·B'	2	0100	4	A	0	B
5	NOT(B)	B'	A'·B' + A·B'	0, 2	0101	5	0	1	B
6	BUF(B)	B	A'·B + A·B	1, 3	1010	6	0	B	1
7	AND(A, B)	A·B	A·B	3	1000	8	0	B	A
8	BUF(A)	A	A·B' + A·B	2, 3	1100	9	0	A	1
9	OR(A, B)	A+B	A'·B + A·B' + A·B	1, 2, 3	1110	13	B	1	A
10	'1'	'1'	A'·B' + A'·B + A·B' + A·B	0, 1, 2, 3	1111	15	1	1	1

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'


A0
A1
M1
0
1
F1
SA
BUF
INV
AND-11
AND1-1
NOR1-1
NOR-11
OR
AND
WHEEL
A two-input MUX can implement these functions, selected by A0, A1, and SA.
M1
0
1
M3
0
1
F
M2
0
1
S3
S0
S1
A, B
M1
WHEEL1
M2
C, D
WHEEL2
M3
0
1
F
S3
S0
S1
The ACT 1 Logic Module can implement these functions.
(a)
(b)

The ACT 1 Logic Module as a Boolean function generator

(a) A 2:1 MUX viewed as a **function wheel**

(b) The ACT 1 Logic Module is two function wheels, an OR gate, and a 2:1 MUX

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

5.1.4 ACT 2 and ACT 3 Logic Modules

- ACT 1 requires 2 LMs per flip-flop: with unknown interconnect capacitance
- 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
- *combinatorial* module implements *combinational* logic (blame MMI for the misuse of terms)
- ACT 2 **S-Module** (**sequential module**) contains a C-Module and a **sequential element**

5.1.5 Timing Model and Critical Path

Keywords and concepts: timing model • deals only with internal logic • estimates delays • before place-and-route step • nondeterministic architecture • find slowest register–register delay or critical path

Example of timing calculations (a rather complex examination of internal module timing):

- 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:

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

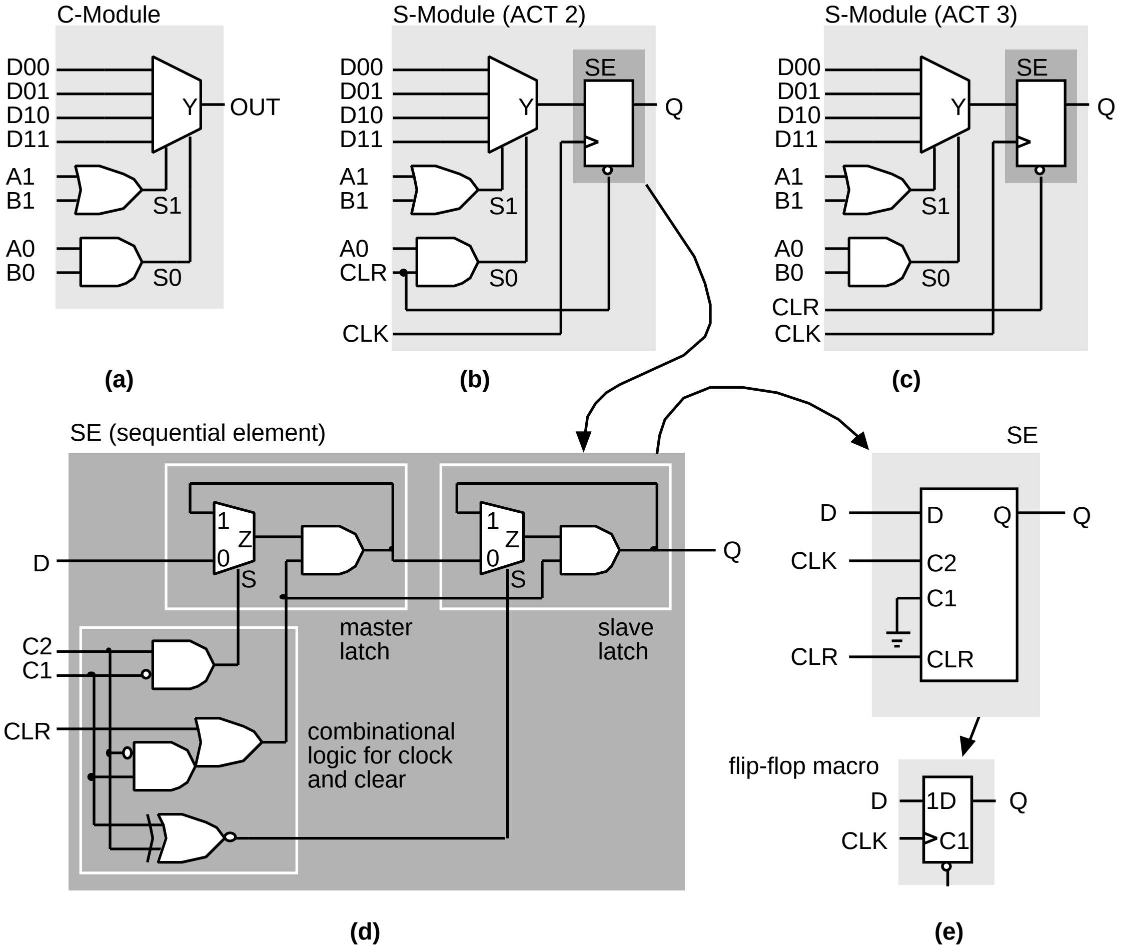
- We do not know the *internal* parameters t'_{SUD}, t'_{H}, and t'_{CO}, but assume reasonable values:

$t'_{SUD}=0.4\text{ns}$, $t'_{H}=0.1\text{ns}$, $t'_{CO}=0.4\text{ns}$.

- t'_{PD} (combinational logic inside the S-Module) is equal to the C-Module delay, so $t'_{PD}=3\text{ns}$ for the ACT 3
- We do not know t'_{CLKD}; assume a value of $t'_{CLKD}=2.6\text{ns}$ (the exact value does not matter)
- Thus the *external* S-Module parameters are: $t_{SUD}=0.8\text{ns}$, $t_{H}=0.5\text{ns}$, $t_{CO}=3.0\text{ns}$
- These are the same as the ACT 3 S-Module parameters (I chose t'_{CLKD} so they would be)
- Of the 3.0ns combinational logic delay: 0.4ns increases the setup time and 2.6ns increases the clock–output delay, t_{CO}
- Actel says that the combinational logic delay is *buried* in the flip-flop setup time. But this is borrowed money—you have to pay it back.

5.1.6 Speed Grading

- **Speed grading** (or **speed binning**) uses a **binning circuit**
- Measure $t_{PD}=(t_{PLH} + t_{PHL})/2$ — and use the fact that properties match across a chip
- Actel speed grades are based on 'Std' speed grade



Actel ACT 2 and ACT 3 Logic Modules

(a) The C-Module for combinational logic

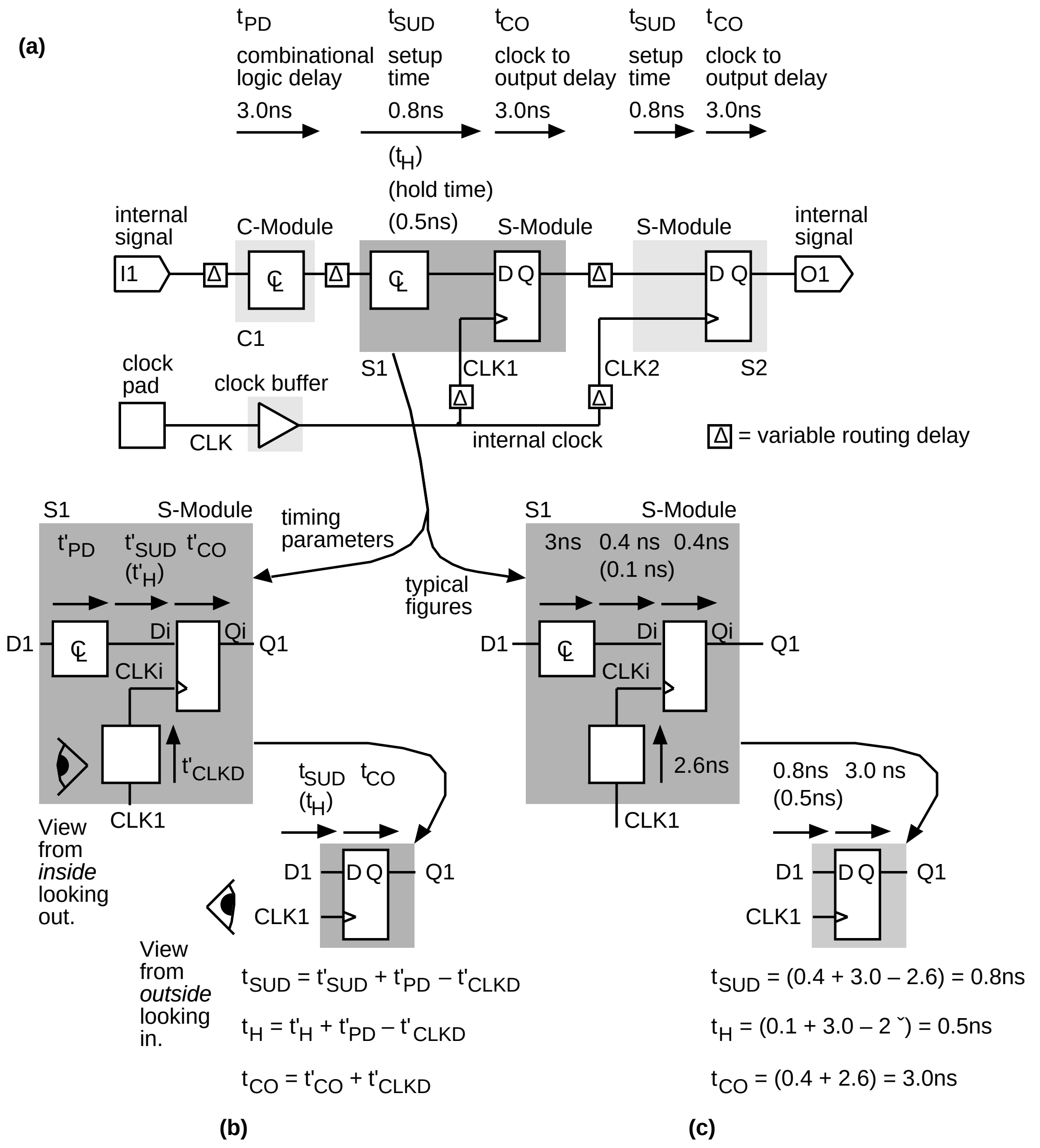
(b) The ACT 2 S-Module

(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

- '1' speed grade is approximately 15 percent faster than 'Std'
- '2' speed grade is approximately 25 percent faster than 'Std'
- '3' speed grade is approximately 35 percent faster than 'Std'.



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

5.1.7 Worst-Case Timing

Keywords and concepts: Using synchronous design you worry about how slow your circuit may be—not how fast • **ambient temperature**, T_A • package **case temperature**, T_C (military) • temperature of the chip, the **junction temperature**, T_J • nominal operating conditions: V_{DD}=5.0V, and T_J=25°C • **worst-case commercial** conditions: V_{DD}=4.75V, and T_J=+70°C • always design using **worst-case timing** • **derating factors** • **critical path delay** between registers • **process corner** (slow–slow • fast–fast • slow–fast • fast–slow) • Commercial. V_{DD}=5V ± 5%, T_A (ambient)=0 to +70°C • Industrial. V_{DD}=5V ± 10%, T_A (ambient)=–40 to +85°C • Military: V_{DD}=5V ± 10%, T_C (case)=–55 to +125°C • Military: Standard MIL-STD-883C Class B • Military extended: unmanned spacecraft

ACT 3 timing parameters

		Fanout				
Family	**Delay**	**1**	**2**	**3**	**4**	**8**
ACT 3-3 (data book)	t_{PD}	2.9	3.2	3.4	3.7	4.8
ACT3-2 (calculated)	t_{PD}/0.85	3.41	3.76	4.00	4.35	5.65
ACT3-1 (calculated)	t_{PD}/0.75	3.87	4.27	4.53	4.93	6.40
ACT3-Std (calculated)	t_{PD}/0.65	4.46	4.92	5.23	5.69	7.38

ACT 3 derating factors

	Temperature T_J (junction)/°C						
V_{DD}/V	**–55**	**–40**	**0**	**25**	**70**	**85**	**125**
4.5	0.72	0.76	0.85	0.90	1.04	1.07	1.17
4.75	0.70	0.73	0.82	0.87	1.00	1.03	1.12
5.00	0.68	0.71	0.79	0.84	0.97	1.00	1.09
5.25	0.66	0.69	0.77	0.82	0.94	0.97	1.06
5.5	0.63	0.66	0.74	0.79	0.90	0.93	1.01

5.1.8 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

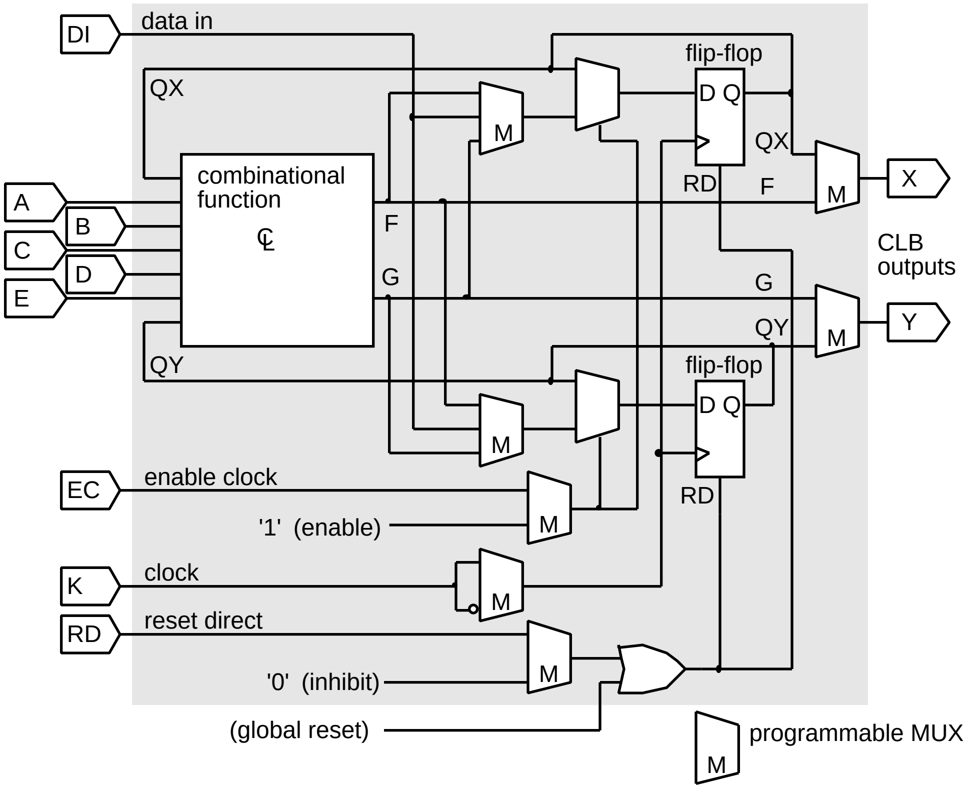
5.2 Xilinx LCA

Keywords and concepts: Xilinx LCA (a trademark, logic cell array) • **configurable logic block** • **coarse-grain architecture**

5.2.1 XC3000 CLB

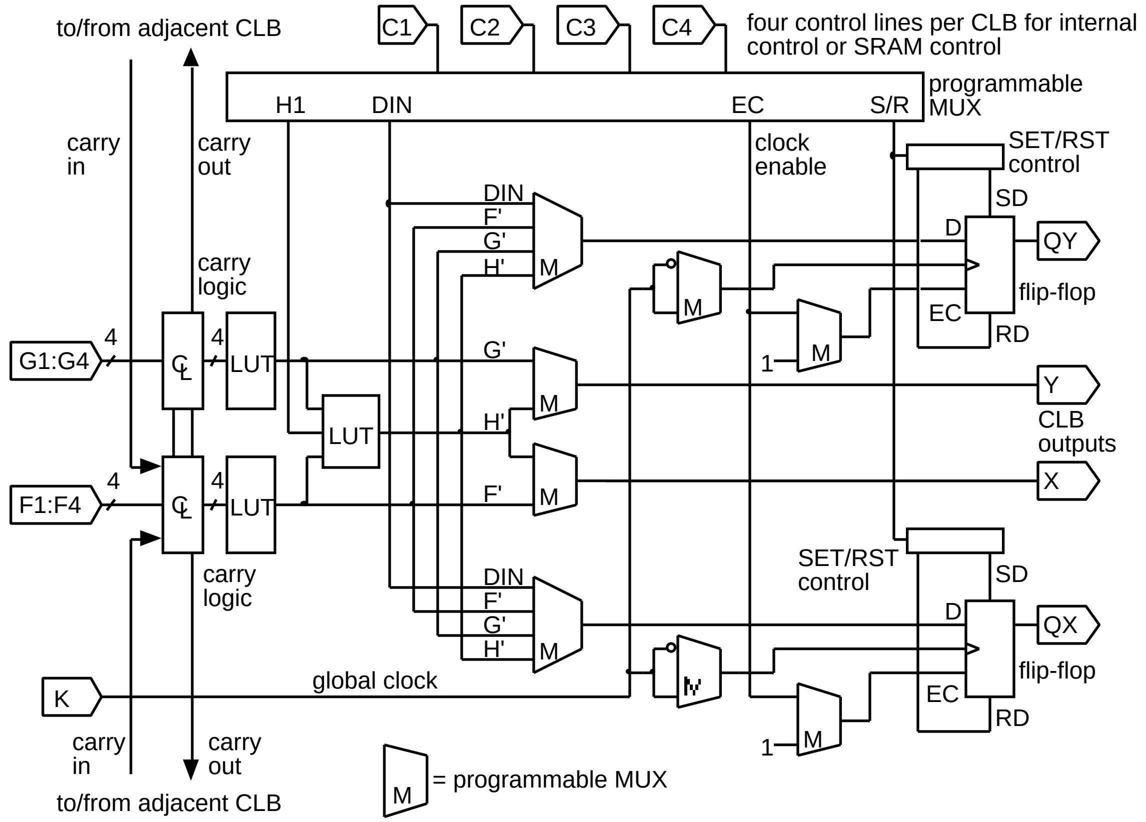
- A 32-bit **look-up table** (LUT)
- CLB propagation delay is fixed (the LUT access time) and independent of the logic function
- 7 inputs to the XC3000 CLB: 5 CLB inputs (A–E), and 2 flip-flop outputs (QX and QY)
- 2 outputs from the LUT (F and G). Since a 32-bit LUT requires only five variables to 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 implemen some functions of 6 and 7 variables).

5.2.2 XC4000 Logic Block



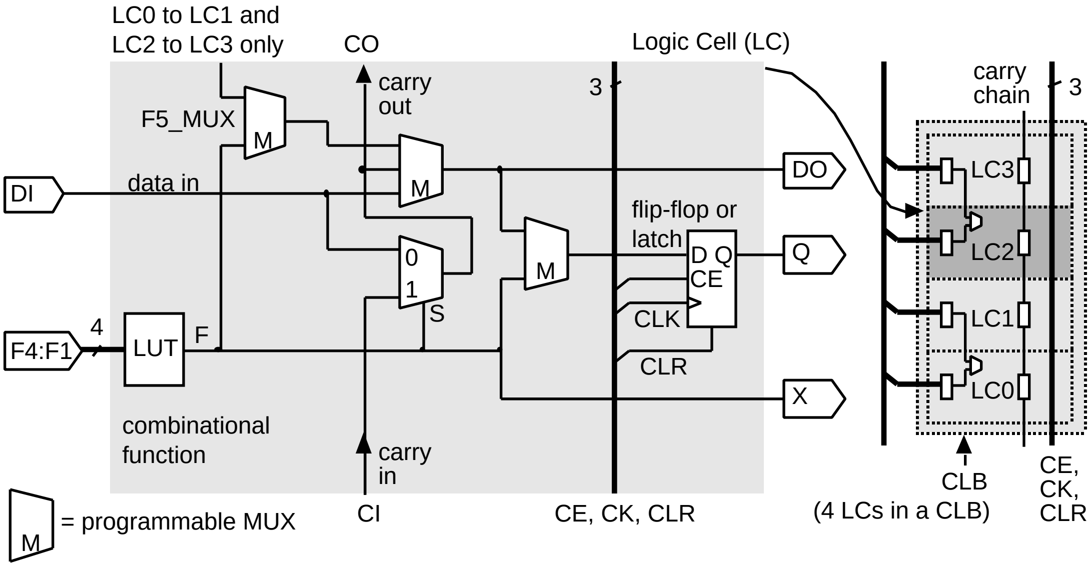
The Xilinx XC3000 CLB (configurable logic block)

(Source: Xilinx.)



The Xilinx XC4000 family CLB (configurable logic block). (*Source:* Xilinx.)

5.2.3 XC5200 Logic Block



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

5.2.4 Xilinx CLB Analysis

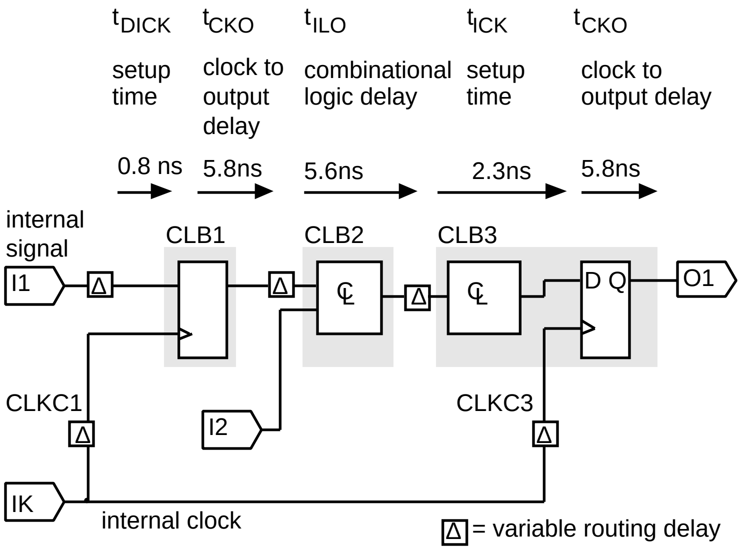
The use of a LUT has advantages and disadvantages:

- An inverter is as slow as a five-input NAND
- A LUT simplifies timing of synchronous logic
- Matched to large SRAM programming technology

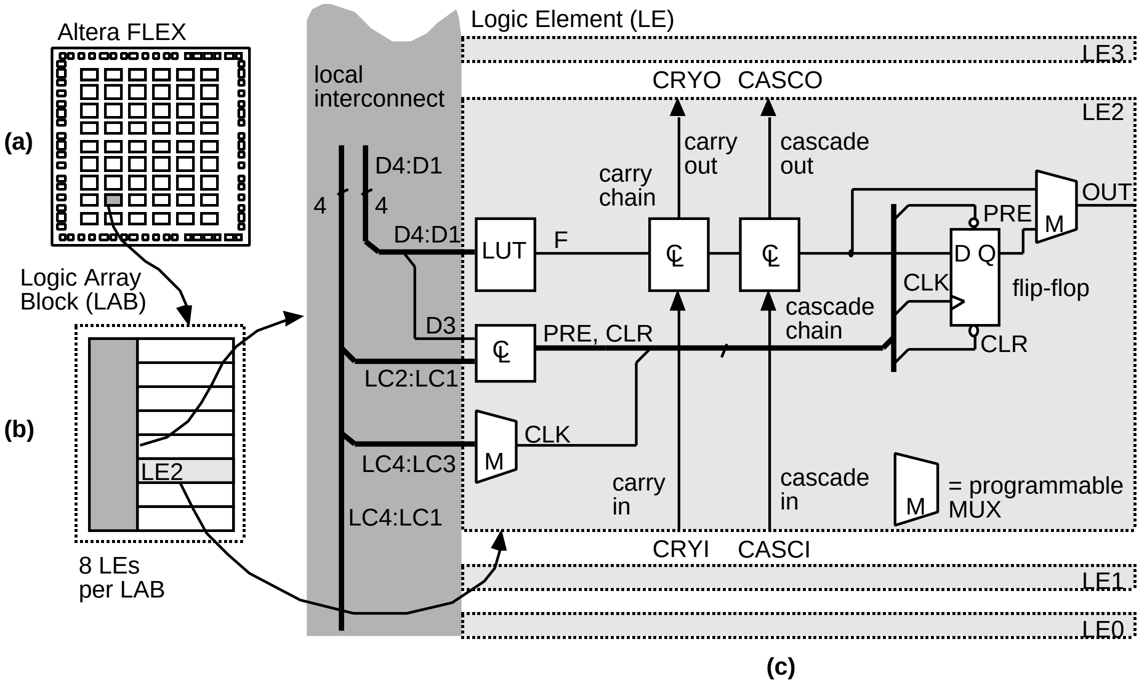
Xilinx uses two speed-grade systems:

- Maximum guaranteed toggle rate of a CLB flip-flop (in MHz) as a suffix—higher is faster
- Example: Xilinx XC3020-125 has a toggle frequency of 125MHz
- Delay time of the combinational logic in a CLB in ns—lower is faster
- Example: XC4010-6 has t_{ILO}=6.0ns
- Correspondence between grade and t_{ILO} is fairly accurate for the XC2000, XC4000, and XC5200 but not for the XC3000

Xilinx LCA timing model (XC5210-6) (*Source:* Xilinx.)



5.3 Altera FLEX



The Altera FLEX architecture

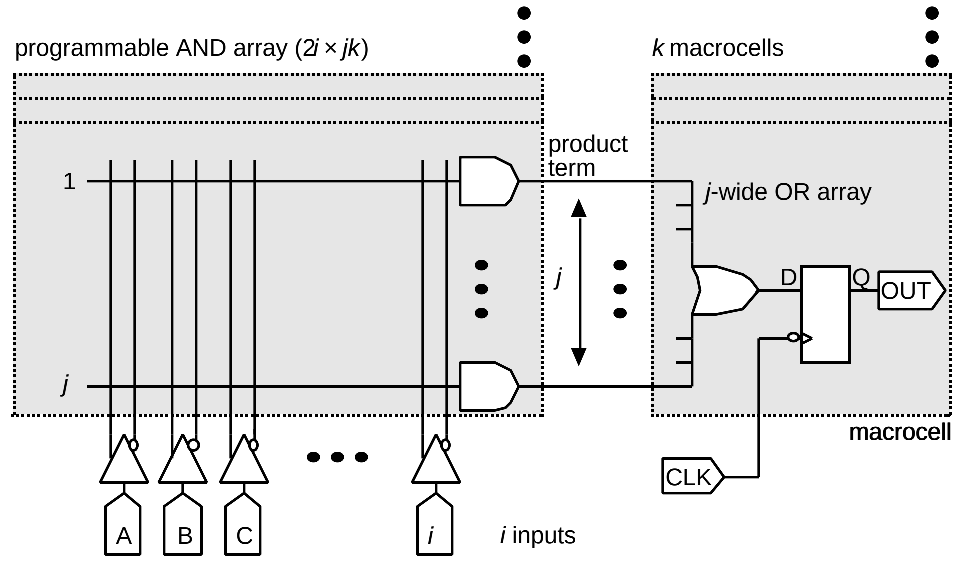
(a) Chip floorplan

(b) Logic Array Block (LAB)

(c) Details of the **Logic Element** (LE)

(*Source:* Altera (adapted with permission).)

5.4 Altera MAX

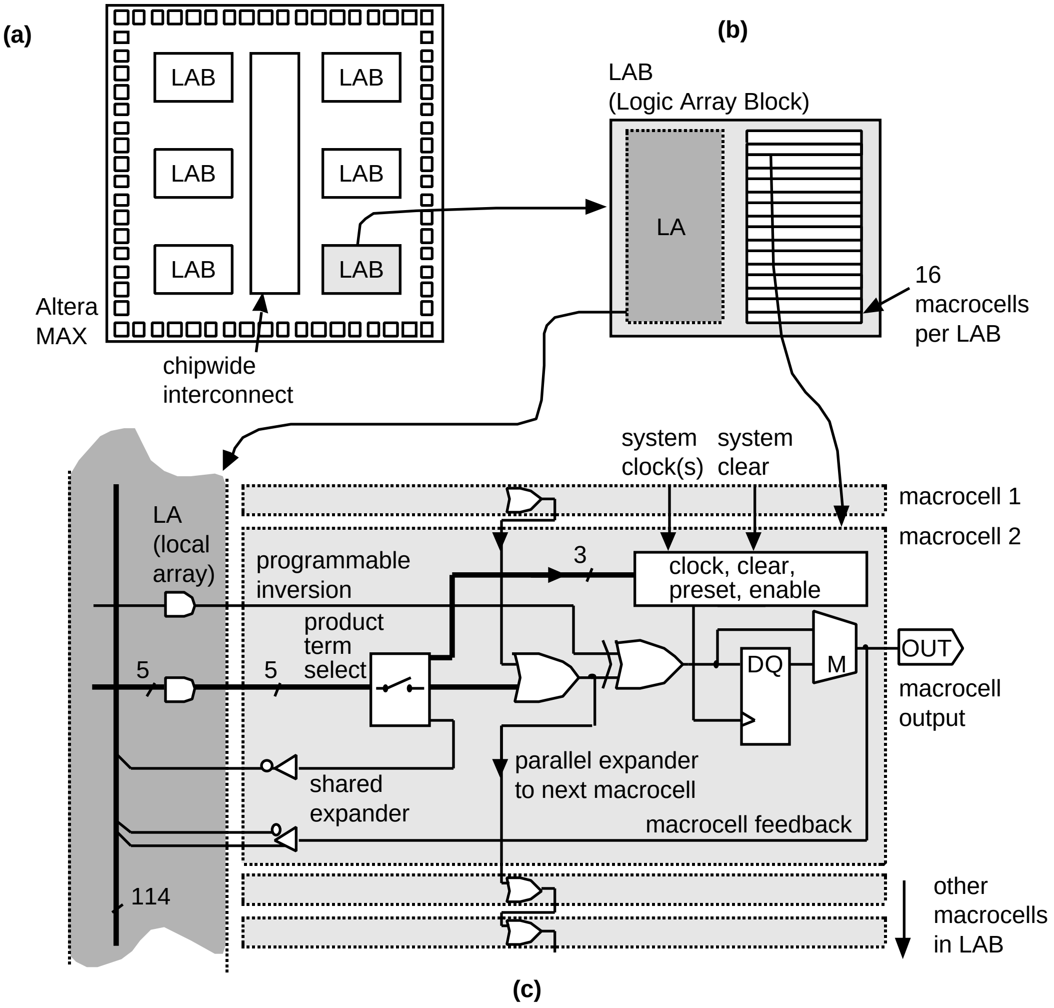


A registered PAL with *i* inputs, *j* product terms, and *k* macrocells. (*Source:* Altera (adapted with permission).)

Features and keywords:

- product-term line
- programmable array logic
- bit line
- word line
- programmable-AND array (or product-term array)
- pull-up resistor
- wired-logic
- wired-AND
- macrocell
- 22V10 PLD

5.4.1 Logic Expanders

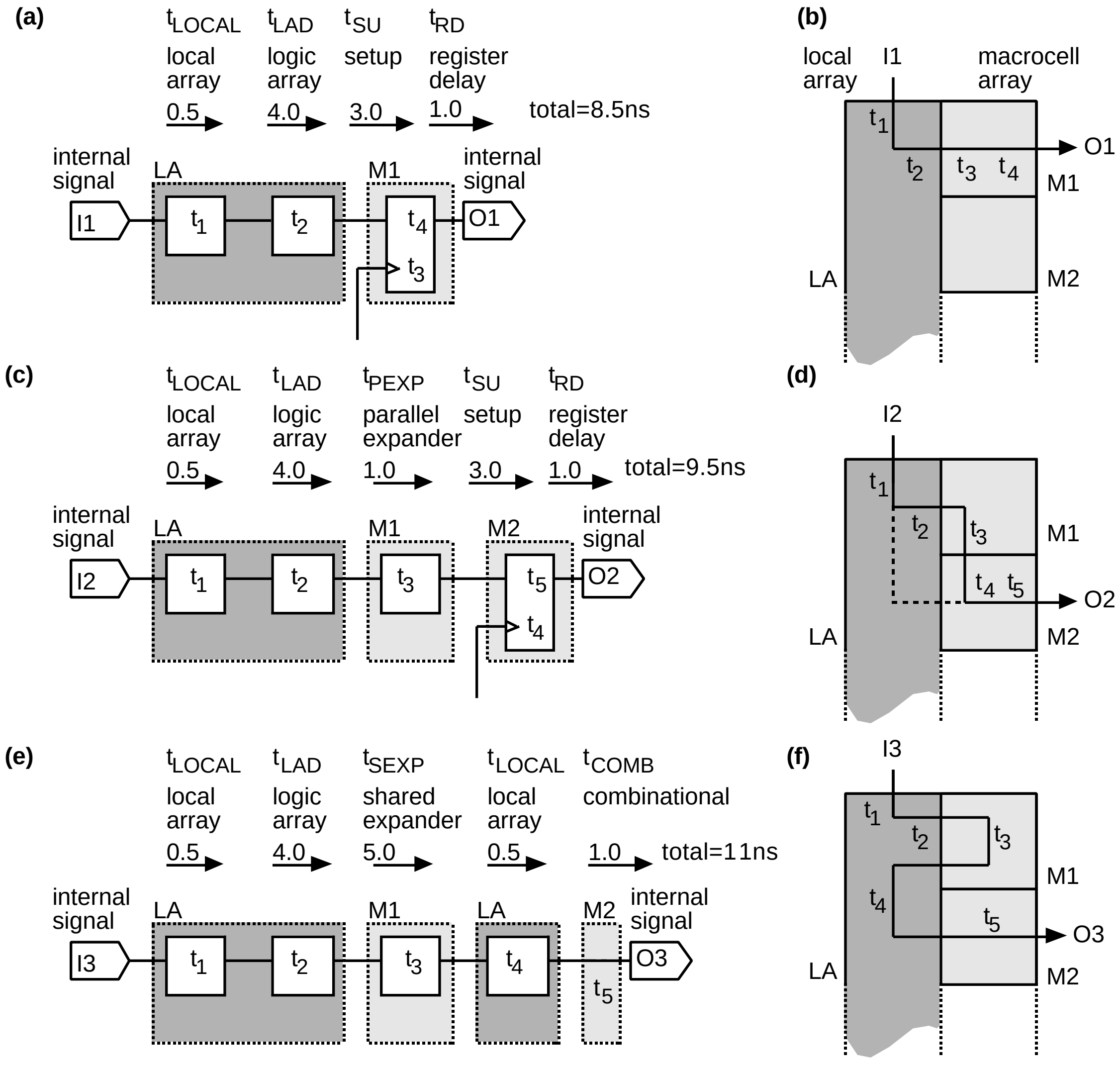


The Altera MAX architecture (the macrocell details vary between the MAX families—the functions shown here are closest to those of the MAX 9000 family macrocells) (*Source:* Altera (adapted with permission).) **(a)** Organization of logic and interconnect **(b)** LAB (Logic Array Block) **(c)** Macrocell

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

5.4.2 Timing Model



Altera MAX timing model (ns for the MAX 9000 series, '15' speed grade) (*Source:* Altera .)

(a) A direct path through the logic array and a register

(b) Timing for the direct path

(c) Using a parallel expander

(d) Parallel expander timing

(e) Making two passes through the logic array to use a shared expander

(f) Timing for the shared expander (there is no register in this path)

5.4.3 Power Dissipation in Complex PLDs

Key points: **static power • Turbo Bit**

5.5 Summary

Key points: The use of multiplexers, look-up tables, and programmable logic arrays • The difference between fine-grain and coarse-grain FPGA architectures • Worst-case timing design • Flip-flop timing • Timing models • Components of power dissipation in programmable ASICs • Deterministic and nondeterministic FPGA architectures

5.6 Problems