Figure 6-1 Layout of Part of a Programmable Logic Cell Array

- **Configurable Logic Block**
- **I/O Block**
- **Interconnect Area**
Figure 6-2 Configuration Memory Cell

WRITE

DATA

Q

\overline{Q}

CONFIGURATION CONTROL
Figure 6-3 Xilinx 3000 Series Logic Cell

DATA IN
QX
QY
LOGIC VARIABLES
A
B
C
D
E

ENABLE CLOCK
EC
CLOCK
K
DIRECT RESET
RD

1 (ENABLE)
0 (INHIBIT)

GLOBAL RESET

COMBINATORIAL FUNCTION
F
G

MUX

CLB OUTPUTS

X
Y

D Q

RD

RD

RD
Figure 6-4 Combinatorial Logic Options

Any Function of ≤ 4 Variables

Any Function of 5 Variables

FG MODE

FGM MODE

A B QX QY
C D E

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

A B QX QY
C D

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

A B QX QY
C D

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

A B QX QY
C D

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

A B QX QY
C D

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

Any Function of ≤ 4 Variables

A B QX QY
Figure 6-5  Flip-flops with Clock Enable

\[ Q^+ = EC \ D1 + EC' \ Q \]

Diagram showing the equivalent circuit for a flip-flop with clock enable.
Figure 6-6 Parallel Adder-Subtractor Logic Cell

\[ F = \text{sum} = a_i^+ = a_i \oplus (b_i \oplus Su) \oplus c_i \]

\[ G = c_{i+1} = \text{carry out} = a_i c_i + (a_i + c_i)(b_i \oplus Su) \]
Figure 6-7 Signal Paths Within Adder-Subtractor Logic Cell

DATA IN (DI)

LOGIC VARIABLES:
- $c_i$
- $b_i$
- $A$
- $B$
- $C$
- $D$
- $E$
- $S_u$

COMBINATORIAL FUNCTION:
- $QX$
- $QY$

ENABLE CLOCK (EC)

DIRECT RESET (RD)

1 (ENABLE)

0 (INHIBIT)

(GLOBAL RESET)

CLB OUTPUTS:
- $a_i$
- $c_{i+1}$

MUX (MLX 1)

INTERNAL CIRCUITRY
Figure 6-9
General-purpose Interconnects

Figure 6-10
Direct Interconnects Between Adjacent CLBs
Figure 6-11 Vertical and Horizontal Long Lines
Figure 6-12 Uses of Tristate Buffers

(a) Multiplexer implementation

(b) Wired-AND implementation
Figure 6-13 Crystal Oscillator
6.2  Designing with FPGAs

Sophisticated CAD tools are available to assist with the design of systems using programmable gate arrays. One method of designing a digital system with a FPGA uses the following steps:

1. Draw a block diagram of the digital system. Define condition and control signals and construct SM charts or state graphs which describe the required sequence of operations.

2. Write a VHDL description of the system. Simulate and debug the VHDL code, and make any necessary corrections to the design which was developed in step 1.

3. Work out the detailed logic design of the system using gates, flip-flops, registers, counters, adders, etc.

4. Enter a logic diagram of the system into the computer using a schematic capture program. Simulate and debug the logic diagram, and make any necessary corrections to the design of step 3.

5. Run a partitioning program. This program will break the logic diagram into pieces which will fit into the configurable logic blocks.

6. Run an automatic place and route program. This will place the logic blocks in appropriate places in the FPGA and then route the interconnections between the logic blocks.

7. Run a program which will generate the bit pattern necessary to program the FPGA.

8. Download the bit pattern into the internal configuration memory cells in the FPGA and test the operation of the FPGA.
Figure 6-14  EPROM Connection for LCA Initialization
State Assignment:
T0: AB = 00, T1: AB = 01, T2: AB = 10, T3: AB = 11

\[ A^+ = A'B' \text{Dn\_roll D711} + A'B' \text{Dn\_roll D2312} + A'B \text{Dn\_roll Eq} + A'B \text{Dn\_roll D7} + A \text{Reset'} \]

\[ B^+ = A'B' \text{Dn\_roll D711'} + A'B \text{Dn\_roll'} + A'B \text{Eq'} + A B \text{Reset'} \]

Win = A B'
Lose = A B
En\_roll = A'

Sp = A'B' \text{Dn\_roll D711'} \text{D2312'}

Q+ = Q' En\_roll Rb + Q Rb
Roll = Q Rb
Dn\_roll = Q Rb'
Figure 6-15 Dice Game Block Diagram
Figure 6-16 Dice Game Controller Module

(a) Main controller
Figure 6-16  Dice Game Controller Module

(b) Dice roll controller
Figure 6-17  Modulo-6 Counter
Figure 6-18 Layout and Routing for Dice Game for XC3020
Figure 6-19a  Realization of General 6-variable Functions

\[ Z(a,b,c,d,e,f) = a'Z(0,b,c,d,e,f) + aZ(1,b,c,d,e,f) = a'Z_0 + aZ_1 \]
Figure 6-19b Realization of General 7-variable Function

\[
Z(a,b,c,d,e,f,g) = a'b'Z(0,0,c,d,e,f,g) + a'bZ(0,1,c,d,e,f,g) + ab'Z(1,0,c,d,e,f,g) + abZ(1,1,c,d,e,f,g)
\]

\[
= a'b'Y_0 + a'bY_1 + ab'Y_2 + abY_3
\]
Figure 6-20  Simplified Block Diagram for 4000 Series CLB

LOGIC FUNCTION OF F1-F4

LOGIC FUNCTION OF G1-G4

LOGIC FUNCTION OF F, G, AND H1

S/R CONTROL

D  SD  Q  YQ

D  SD  Q  XQ

X

Y
Figure 6-21  XC4000 Dedicated Carry Logic
Figure 6-22
Conceptual Diagram of a Typical Addition (2 Bits/CLB)
Figure 6-23 Connections for a 4-bit Adder
Figure 6-24  CLB as a Read/Write Memory Cell
Figure 6-25  4000 Series I/O Block
library BITLIB;
use BITLIB.bit_pack.all;

entity XC4000CLB is
    port (MEM_BITS : in bit_vector(0 to 51);
        G_IN, F_IN, C_IN : in bit_vector(4 downto 1);
        K : in bit; Y,X : out bit; Q : out bit_vector (1 downto 0));
end XC4000CLB;

architecture behavior of XC4000CLB is
    alias G_FUNC : bit_vector(0 to 15) is MEM_BITS(0 to 15);
    alias F_FUNC : bit_vector(0 to 15) is MEM_BITS(16 to 31);
    alias H_FUNC : bit_vector(0 to 7) is MEM_BITS(32 to 39);
    type bv2D is array (1 downto 0) of bit_vector(1 downto 0);
    constant FF_SEL : bv2D := (MEM_BITS(40 to 41),MEM_BITS(42 to 43));
    alias Y_SEL : bit is MEM_BITS(44);
    alias X_SEL : bit is MEM_BITS(45);
    alias EDGE_SEL: bit_vector(1 downto 0) is MEM_BITS(46 to 47);
    alias EC_SEL : bit_vector(1 downto 0) is MEM_BITS(48 to 49);
    alias SR_SEL : bit_vector(1 downto 0) is MEM_BITS(50 to 51);
    alias H1 : bit is C_IN(1); alias DIN : bit is C_IN(2);
    alias SR : bit is C_IN(3); alias EC : bit is C_IN(4);

    -- Timing spec for XC4000, Speed Grade -4
    constant Tiho : TIME := 6 ns; -- F/G inputs to X/Y outputs via H
    constant Tilo : TIME := 4 ns; -- F/G inputs to X/Y outputs
    constant Tcko : TIME := 3 ns; -- Clock K to Q outputs
    constant Trio : TIME := 7 ns; -- S/R to Q outputs
    signal G,F,H : bit;
begin
  G <= G_FUNC (vec2int(G_IN));  F <= F_FUNC (vec2int(F_IN));
  H <= H_FUNC (vec2int(H1&G&F)) after (Tiho-Tilo);
  X <= (X_SEL and H) or (not X_SEL and F) after Tilo;
  Y <= (Y_SEL and H) or (not Y_SEL and G) after Tilo;
process (K, SR) -- update FF outputs
variable DFF_EC,D : bit_vector(1 downto 0);
begin
  for i in 0 to 1 loop
    DFF_EC(i) := EC or EC_SEL(i);
    case FF_SEL(i) is
    when "00" => D(i) := DIN;
    when "01" => D(i) := F;
    when "10" => D(i) := G; when "11" => D(i) := H;
    end case;
  if (SR='1') then Q(i)<=SR_SEL(i) after Trio; -- If SR set, then set or reset ff
  else
    if (DFF_EC(i)='1') then -- If clock enabled then
      -- If correct triggering edge then update ff value
      if (((EDGE_SEL(i)='1' and rising_edge(K)) or (EDGE_SEL(i)='0'
          and falling_edge(K))) then Q(i)<=D(i) after Tcko; end if;
    end if;
  end if;
  end loop;
end process;
end behavior;
Table 6-1  Truth Tables for $G$ and $F$ Function Generators

<table>
<thead>
<tr>
<th></th>
<th>G4</th>
<th>G3</th>
<th>G2</th>
<th>G1</th>
<th></th>
<th>G4</th>
<th>F3</th>
<th>F2</th>
<th>F1</th>
<th>F0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>M</td>
<td>Q1</td>
<td>Q0</td>
<td>Q1+</td>
<td></td>
<td>M</td>
<td>Q1</td>
<td>Q0</td>
<td>Q0+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
**Figure 6-27 XC4000 Implementation of Multiplier Control**

```vhdl
entity Fig_4_6 is
    port (St, K, M, CLK : in bit; Ad, Sh, Load, Done : out bit);
end Fig_4_6;

architecture CLBs of Fig_4_6 is
    component XC4000CLB
        port(MEM_BITS : in bit_vector(0 to 51);
            G_IN, F_IN, C_IN : in bit_vector(4 downto 1);
            K : in bit; Y,X : out bit; Q : out bit_vector (1 downto 0));
    end component;

    constant MEM1 : bit_vector (0 to 51) :=
        "0000010001100110011000101110101000000000100100110000";
    constant MEM2 : bit_vector (0 to 51) :=
        "000100010001000100000000100010000000000000000000110000";
    constant MEM3 : bit_vector (0 to 51) :=
        "000000000100011001100010000000000000000000000000110000";

    signal Q : bit_vector (1 downto 0);
    signal G_IN1,G_IN2,G_IN3,F_IN1,F_IN2,F_IN3 : bit_vector (3 downto 0);

begin
    G_IN1<=K&M&Q; F_IN1<=St&M&Q; G_IN2<="00"&Q; F_IN2<=St&'0'&Q;
    G_IN3<=M&'0'&Q; F_IN3<=M&'0'&Q;

    CLB1: XC4000CLB port map (MEM1,G_IN1,F_IN1,"1000",CLK,open,open,Q);
    CLB2: XC4000CLB port map (MEM2,G_IN2,F_IN2,"1000",CLK,Done,Load,open);
    CLB3: XC4000CLB port map (MEM3,G_IN3,F_IN3,"1000",CLK,Ad,Sh,open);
end CLBs;
```
Figure 6-28  Partial State Graph

One-hot state assignment for flip-flops Q0 Q1 Q2 Q3:
  \( T0: 1000, \ T1: 0100, \ T2: 0010, \ T3: 0001 \)

\[
\begin{align*}
  Q3^+ &= X1 \ Q0 + X2 \ Q1 + X3 \ Q2 + X4 \ Q3 \\
  Z1 &= X1 \ Q0 + X3 \ Q2 \\
  Z2 &= X2 \ Q1 + X4 \ Q3
\end{align*}
\]
Figure 6-29  Altera 7000 Series Architecture for EPM7032, 7064, and 7096 Devices
Logic Array

Parallel Logic Expanders (from other macrocells)

Global Clear
Global Clock

Product-Term Select Matrix

36 Signals from PIA

16 Expander Product Terms

Shared Logic Expanders

Programmable Register

Register Bypass

Clock/Enable Select

VCC

Clear Select

PRN

Q

EN

CLRN

to I/O Control Block

to PIA

Figure 6-30
Macrocell for EMP7032, 7064, and 7096 Devices
Figure 6-31 Sharable Expanders

36 Signals from PIA
16 Shared Expanders
Figure 6-32 Parallel Expanders

36 Signals from PIA
16 Shared Expanders

from Previous Macrocell

Product-Term Select Matrix

Preset

Clock

Clear

Macrocell Product-term Logic

Macrocell Product-term Logic

to Next Macrocell
Figure 6-33  I/O Block for EPM7032, 7064, and 7096
Figure 6-34 FLEX 10K Device Block Diagram

- **I/O Element (IOE)**
- **Logic Array Block (LAB)**
- **Embedded Array Block (EAB)**
- **Logic Element (LE)**
- **Local Interconnect**

Legend:
- IOE (I/O Element)
- EAB (Embedded Array Block)
- LAB (Logic Array Block)
- Logic Element (LE)
- Local Interconnect
Figure 6-35  FLEX 10K Logic Array Block
Figure 6-36  FLEX 10K Logic Element
Figure 6-37  Cascade Chain Operation

AND Cascade Chain

OR Cascade Chain
Figure 6-38  FLEX 10K Embedded Array Block

- Dedicated Inputs & Global Signals
- Device-Wide Clear
- Row Interconnect
- Column interconnect

- EAB Local Interconnect

- RAM/ROM: 256 x 8, 512 x 4, 1,024 x 2, 2,048 x 1

- Data In, Data Out

- Address

- WE

- 8, 4, 2, 1

- 8, 9, 10, 11

- 2, 4, 8, 16

- 24