Designing with FPGAs

ELEC 418

Advanced Digital Systems

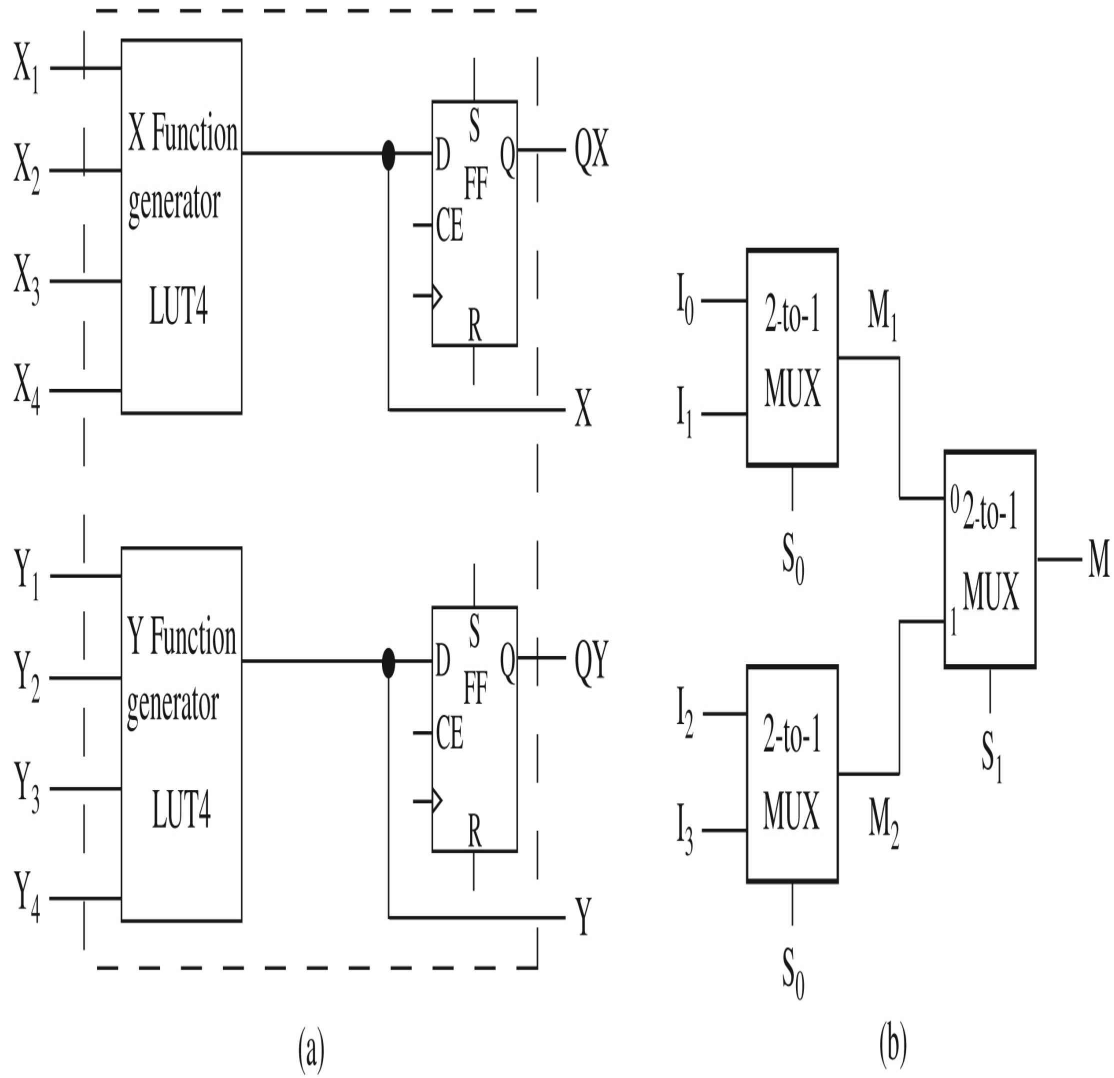
Dr. Ron Hayne

Images Courtesy of Thomson Engineering

THE CITADEL
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

FPGA Logic Blocks

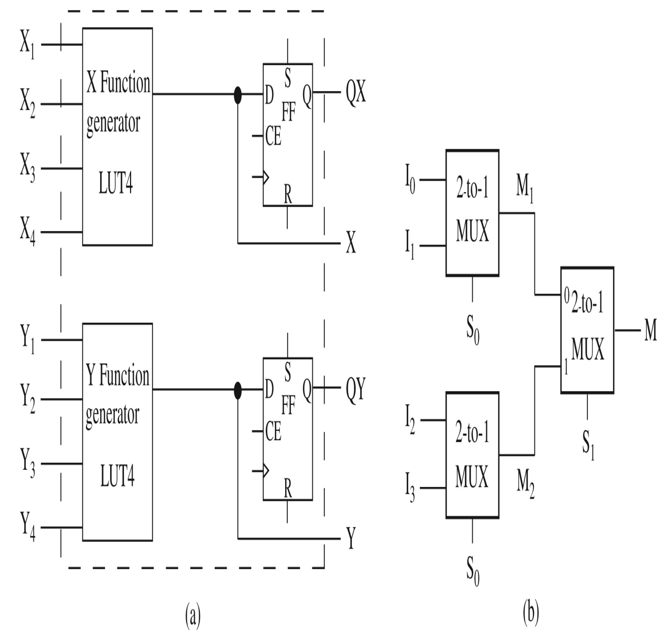
FIGURE 6-1:
(a) Example Building Block for an FPGA; (b) 4-to-1 Multiplexer Using 2-to-1 Multiplexers



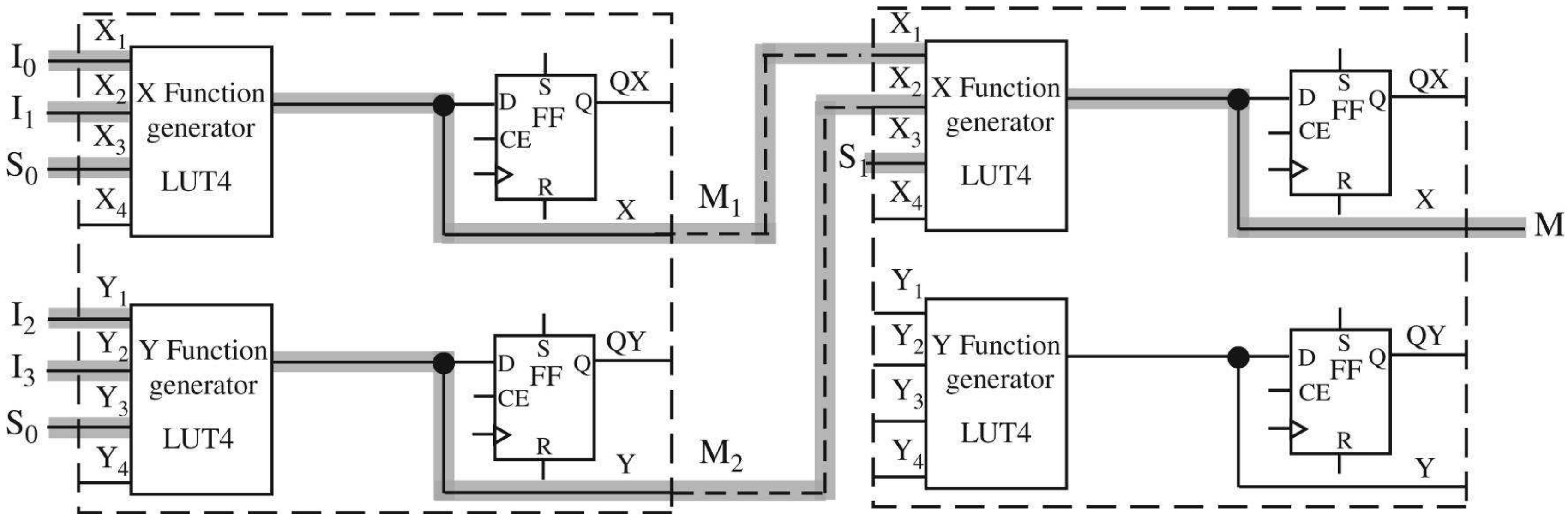
Implementing Functions

- **4-to-1 Multiplexer**
  - $M = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$
- **Decomposition into 2-to-1 Multiplexers**
  - $M_1 = S_0'I_0 + S_0I_1$
  - $M_2 = S_0'I_2 + S_0I_3$
  - $M = S_1'M_1 + S_1M_2$

FIGURE 6-1: (a) Example Building Block for an FPGA; (b) 4-to-1 Multiplexer Using 2-to-1 Multiplexers



Mapping to Logic Blocks



LUT Contents

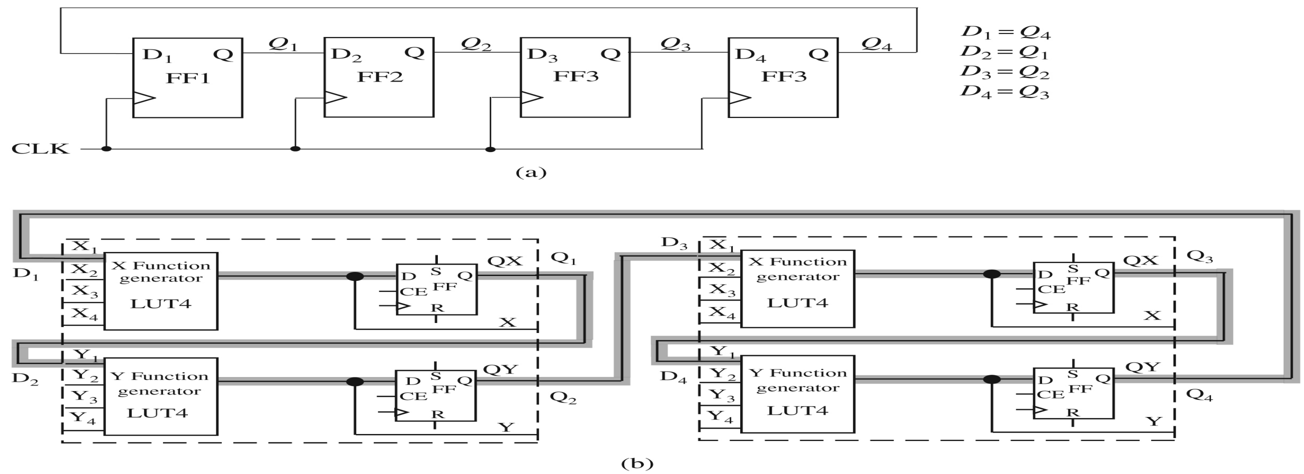
Inputs				Output
X_4	X_3 (S_0)	X_2 (I_1)	X_1 (I_0)	X (M_1)
X	0	0	0	0
X	0	0	1	1
X	0	1	0	0
X	0	1	1	1
X	1	0	0	0
X	1	0	1	0
X	1	1	0	1
X	1	1	1	1

LUT-M1 = 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1

Another Example

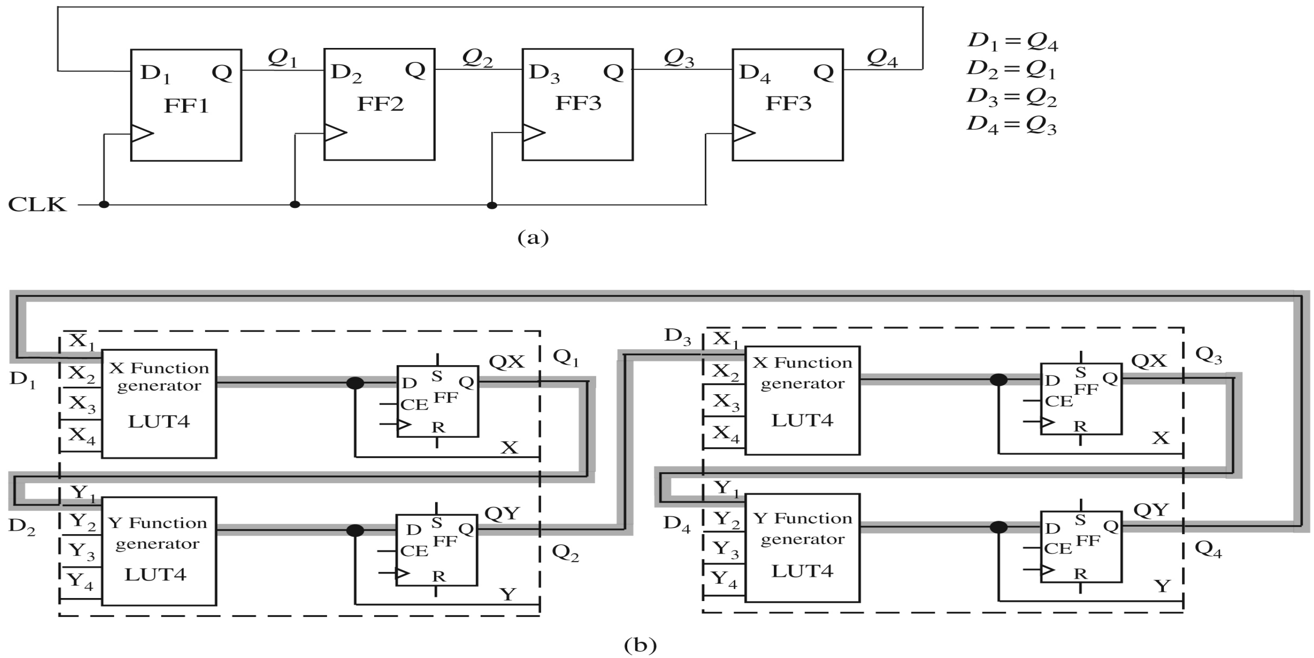
- **Ring Counter**

FIGURE 6-5:
(a) Circular Shift Register;
(b) Implementation Using Simple FPGA Building Block

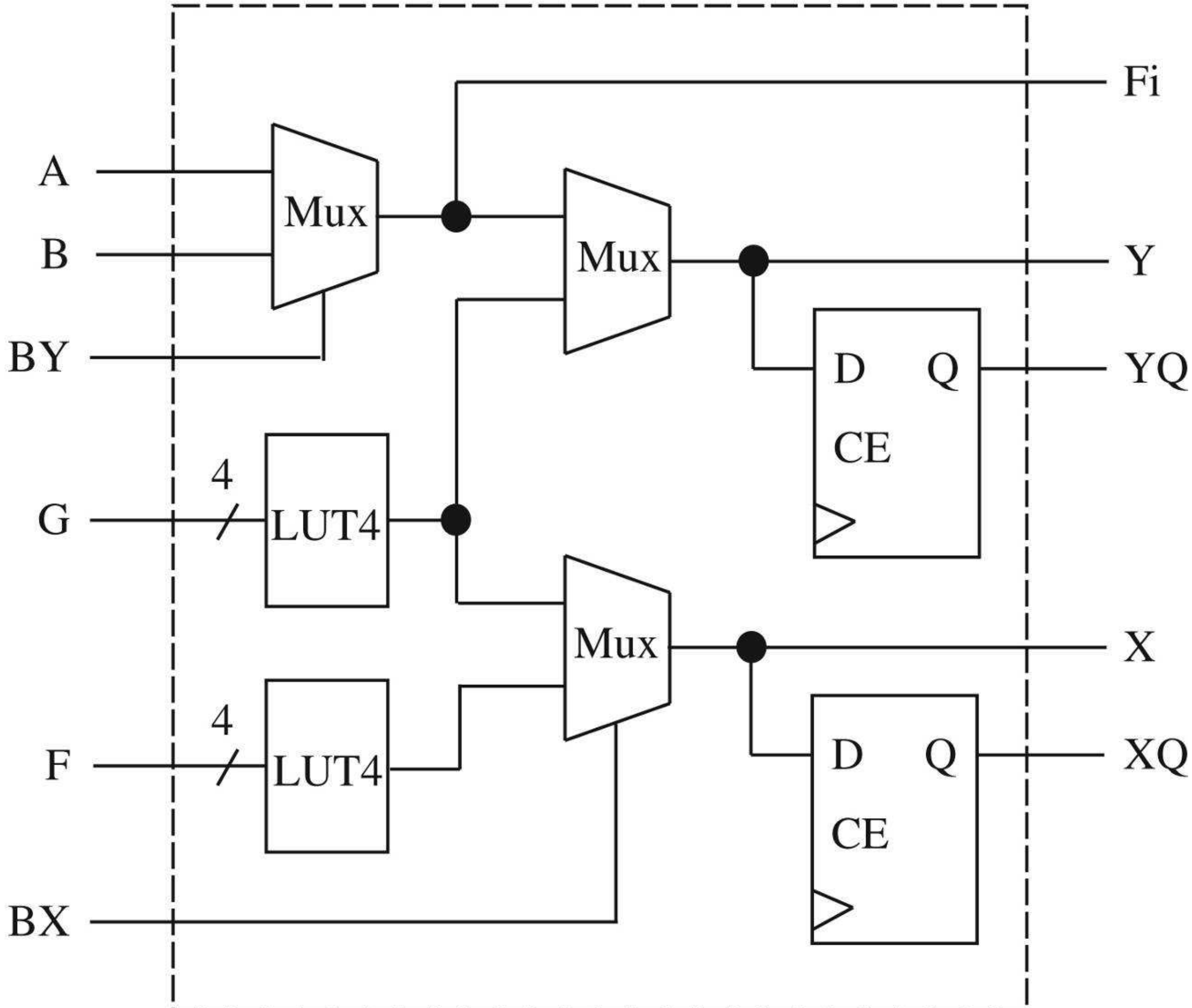


FPGA Implementation

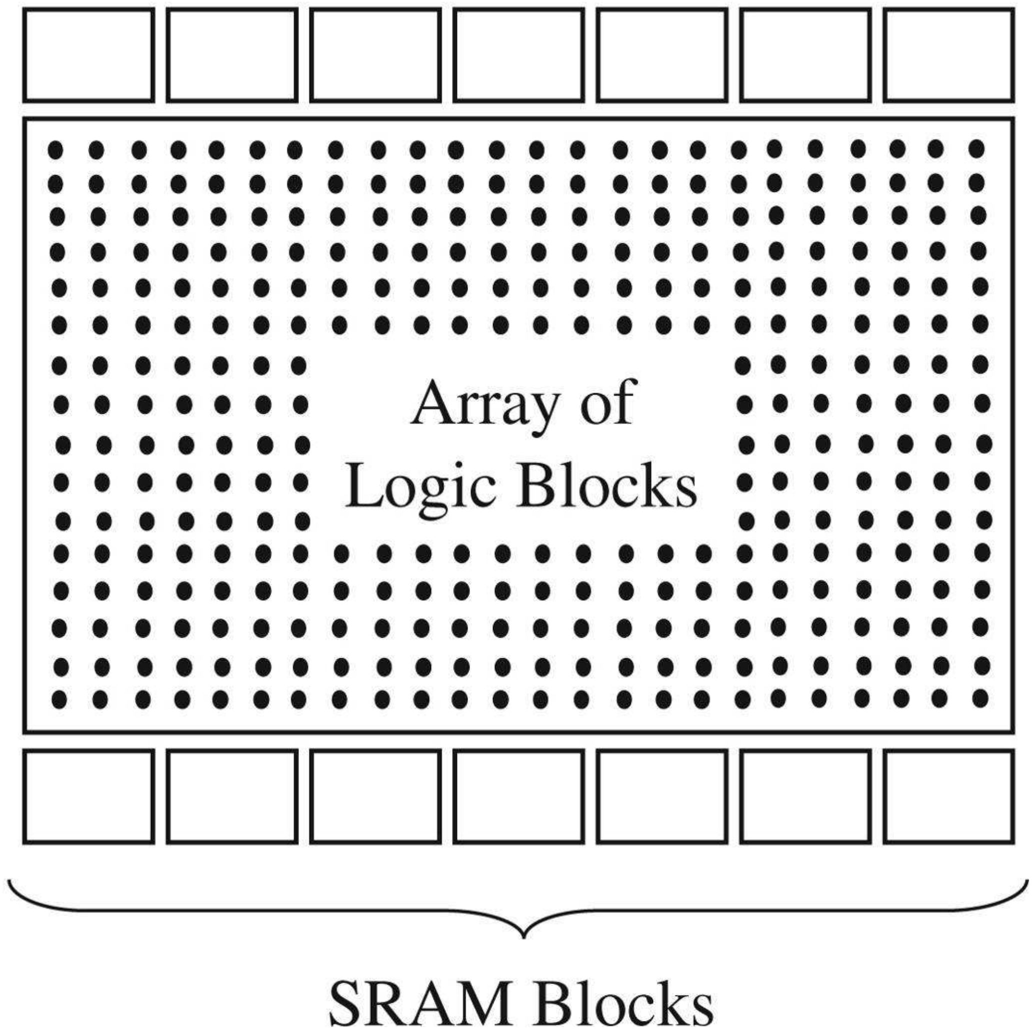
FIGURE 6-5:
(a) Circular Shift Register;
(b) Implementation Using Simple FPGA Building Block



Xilinx Configurable Logic Block



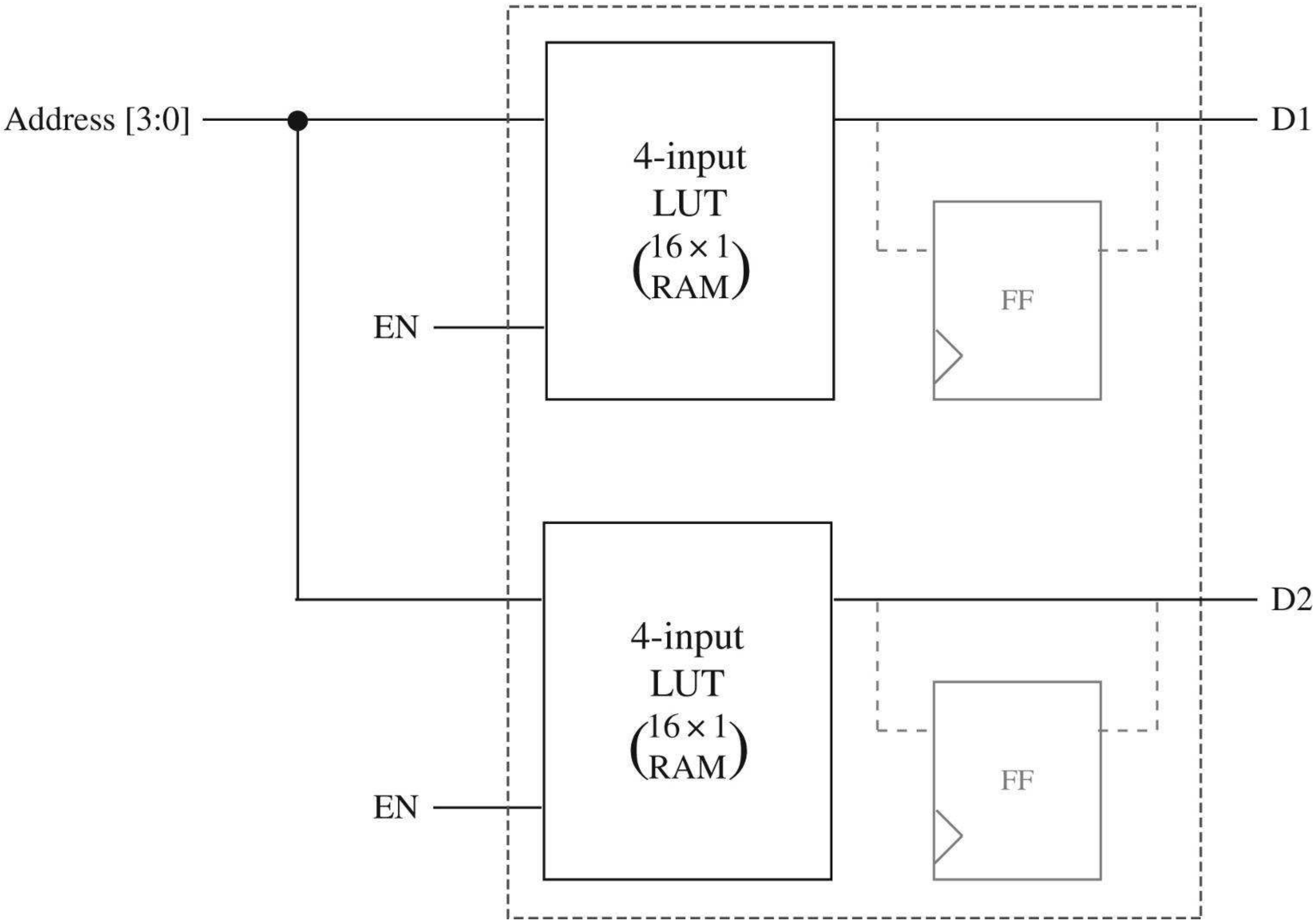
Dedicated Memory in FPGAs



Example RAM Sizes

FPGA Family	Dedicated RAM Size (Kb)	Organization
Xilinx Virtex 5	1152–10368	64–576 18Kb blocks
Xilinx Virtex 4	864–9936	48–552 18Kb blocks
Xilinx Virtex-II	72–3024	4–168 18Kb blocks
Xilinx Spartan 3E	72–648	4–36 18Kb blocks
Altera Stratix II	409–9163	104–930 512b blocks 78–768 4Kb blocks 0–9 512Kb blocks
Altera Cyclone II	117–1125	26–250 4Kb blocks
Lattice SC	1054–7987	56–424 18Kb blocks
Actel Fusion	27–270	6–60 4Kb blocks

Memory From LUTs



VHDL Models for Memory

- **Synchronous or Asynchronous**
- **Synchronous-Write, Asynchronous-Read**
  - LUT-Based Memory
- **Synchronous-Write, Synchronous-Read**
  - Dedicated (Block) Memory

VHDL Models for Memory

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Memory is
 port(Address: in  STD_LOGIC_VECTOR(6 downto 0);
      Clk, MemWrite: in  STD_LOGIC;
      Data_In: in  STD_LOGIC_VECTOR(31 downto 0);
      Data_out: out  STD_LOGIC_VECTOR(31 downto 0));
end Memory;
```

LUT-Based Memory

```
architecture LUT of Memory is
  type RAM is array (0 to 127) of
              std_logic_vector(31 downto 0);
  signal DataMEM: RAM;
begin
  process(CLK)
  begin
    if rising_edge(CLK) then
      if MemWrite = '1' then
        DataMEM(conv_integer(Address)) <= Data_In;
      end if;
    end if;
  end process;
  Data_Out <= DataMEM(conv_integer(Address));
end LUT;
```

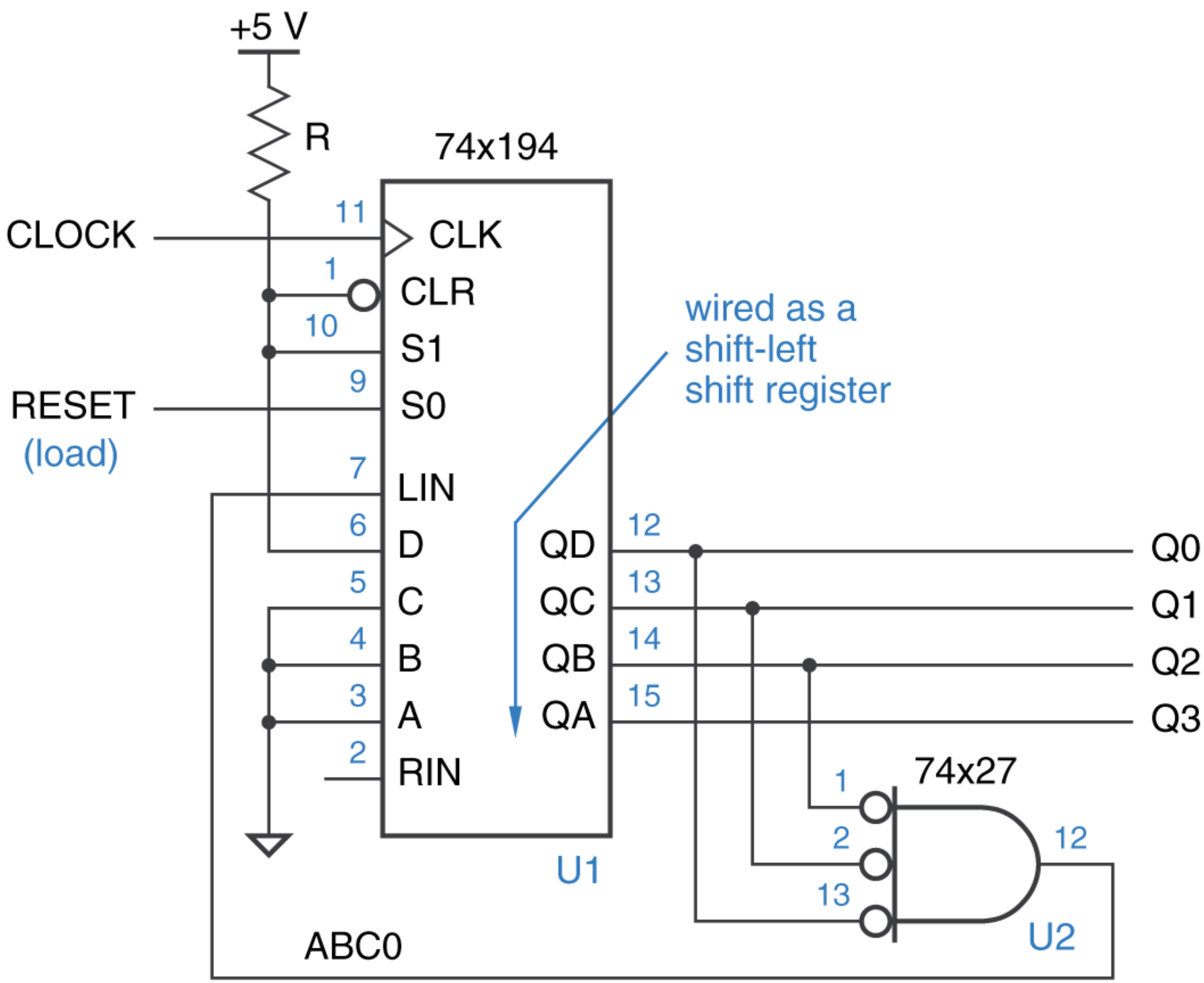
Dedicated Memory

```
architecture Dedicated of Memory is
  type RAM is array (0 to 127) of
              std_logic_vector(31 downto 0);
  signal DataMEM: RAM;
begin
  process(CLK)
  begin
    if rising_edge(CLK) then
      if MemWrite = '1' then
        DataMEM(conv_integer(Address)) <= Data_In;
      end if;
      Data_Out <= DataMEM(conv_integer(Address));
    end if;
  end process;
end Dedicated;
```

CAD Design Flow

- **`Synthesis (Translation)`**
  - Logic Optimization
- **`Mapping`**
- **`Placement`**
- **`Routing`**

Self-Correcting Ring Counter



Synthesis Example

```
-- 4-std_logic Self-Correcting Ring Counter
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RING_COUNT is
  port (CLK, RESET: in std_logic;
        Q : out std_logic_vector(3 downto 0));
end RING_COUNT;

architecture BEHAVE of RING_COUNT is
  signal IQ : std_logic_vector(3 downto 0);
  signal LIN : std_logic;
```

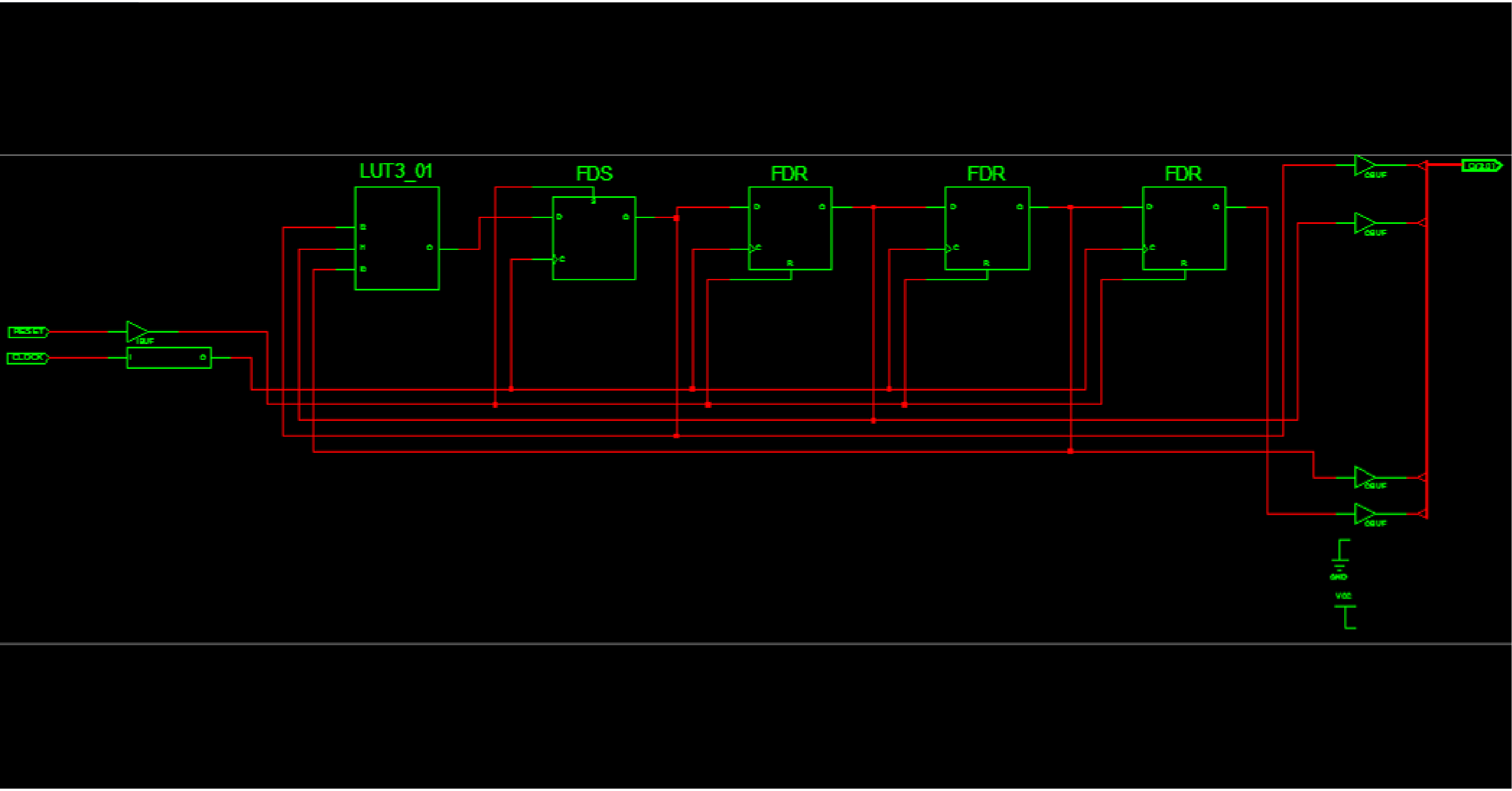
Synthesis Example

```
begin
  LIN <= not IQ(2) and not IQ(1) and not IQ(0);
  process(CLK)
  begin
    if rising_edge(CLK) then
      if RESET = '1' then
        IQ <= "0001";
      else
        IQ <= IQ(2 downto 0) & LIN;
      end if;
    end if;
  end process;
  Q <= IQ;
end BEHAVE;
```

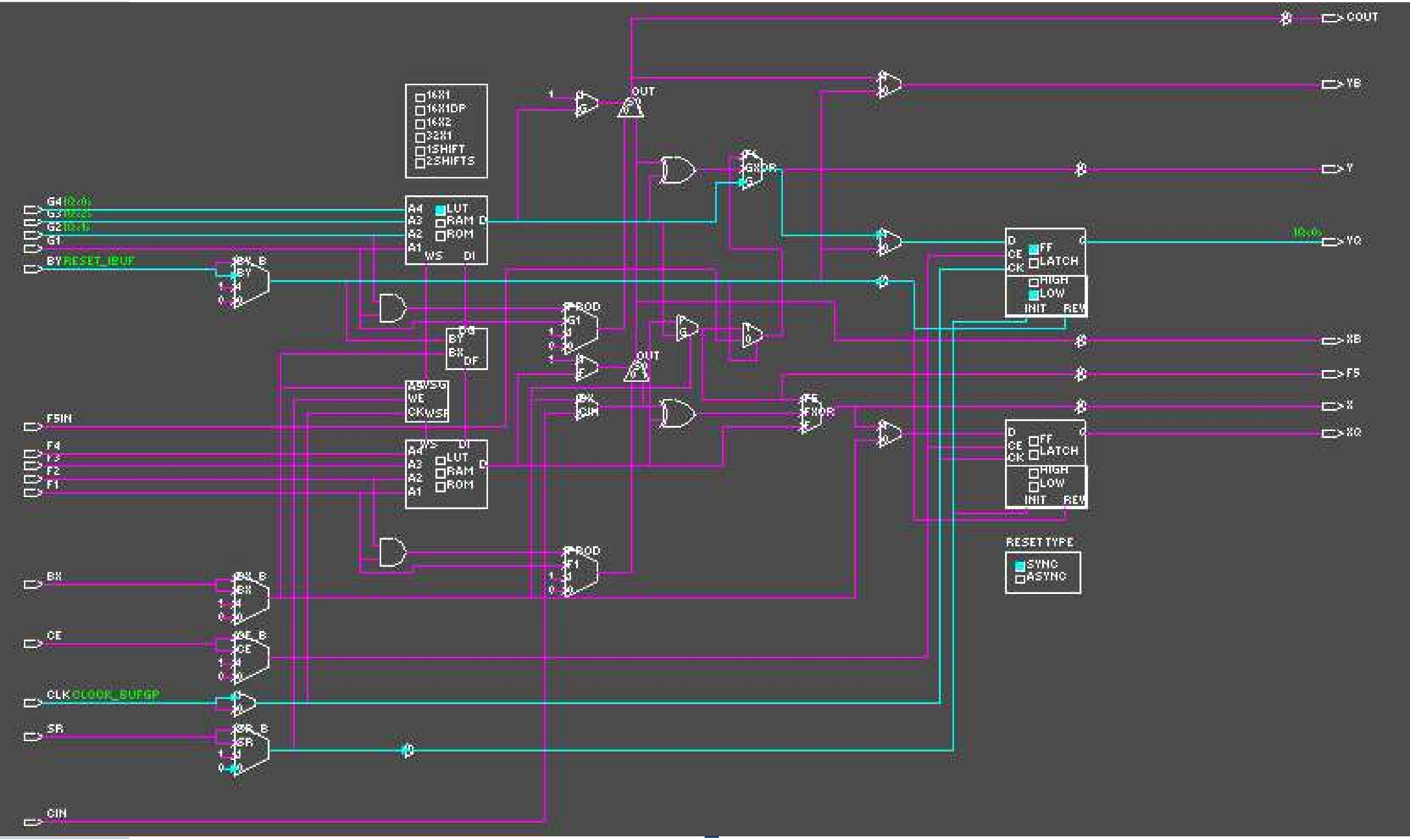
Design Flow Continued

- **Mapping**
  - Process of binding technology-dependent circuits of target technology to technology-independent circuits in the design
    - MUX, ROM, LUT, NAND, NOR
- **Placement**
  - Process of taking defined logic and I/O blocks and assigning them to physical locations
- **Routing**
  - Process of interconnecting the sub-blocks

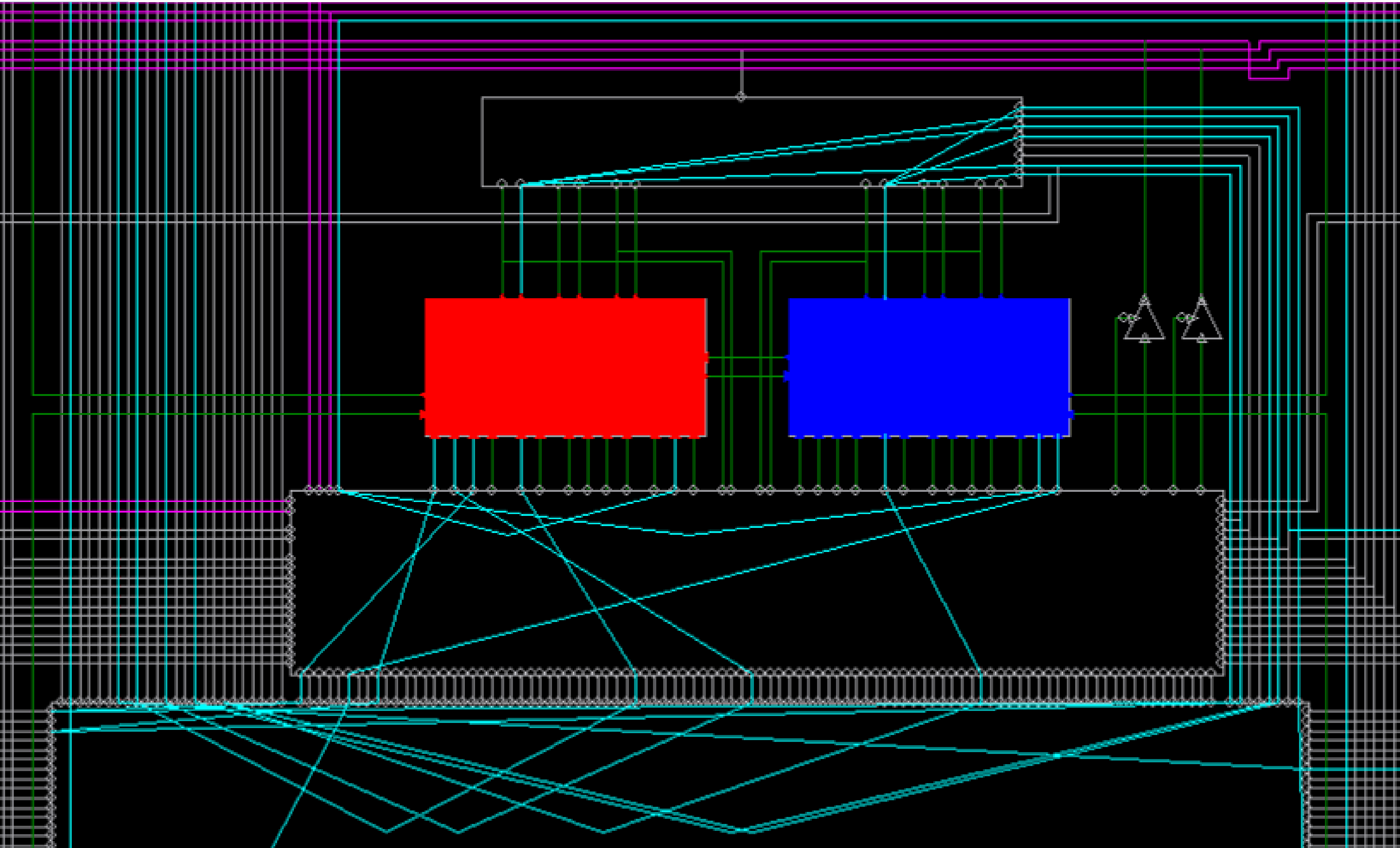
Mapping



Placement


COUT
YB
Y
YQ
XB
F5
X
XQ
G4
G3
G2
G1
BY RESET_IBUF
F5IN
F4
F3
F2
F1
BX
CE
CLK CLOCK_BUFGP
SR
CIN
16X1
16X1DP
16X2
32X1
1SHIFT
2SHIFTS
LUT
RAM
ROM
FF
LATCH
HIGH
LOW
INIT
RESET TYPE
SYNC
ASYNC

Routing



Summary

- **Designing with FPGAs**
  - Implementing Functions
  - Memory
  - CAD Design Flow