# 6.1 Implementing Functions in FPGAs

## 1 Issues Related to Implementing Designs in FPGAs

- hand-mapping simple designs into FPGA building blocks to illustrate tradeoffs arising from the structure of the basic FPGA building blocks.
- Shannon's expansion for decomposition of large functions into smaller functions.
- One-hot state assignment method for FPGA-like technology
- Design flow, synthesis, mapping, and placement issues
- features of some commercial FPGAs

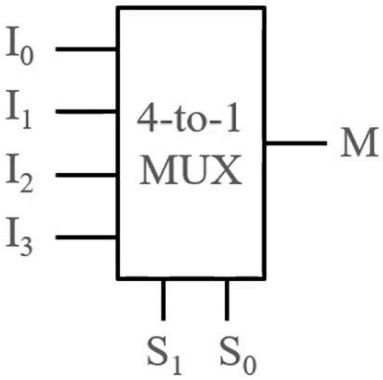
## 2 Implementing Functions in FPGAs

- Many modern FPGAs use a 4-input look-up table (LUT4) as basic building blocks.
- It takes $2^4 = 16$ bits of SRAM to realize LUT4 using SRAM technology.

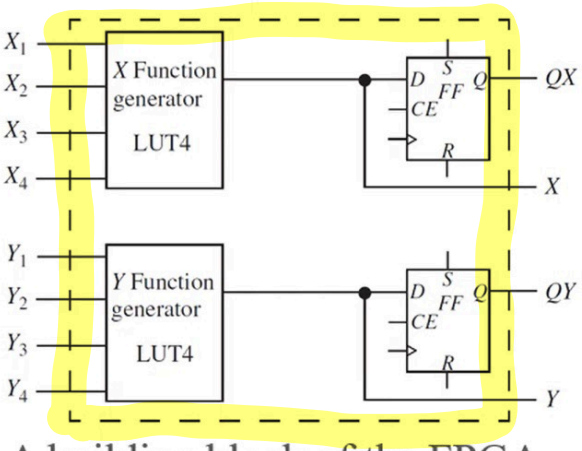
Ex. 4 x 1 MUX

# Example: 4-to-1 MUX

- Design a 4-to-1 multiplexer using an FPGA with ***look-up tables*** (***LUTs***) and ***flip-flops***.
  - Each building block contains two 4-variable function generators, and two flip-flops.



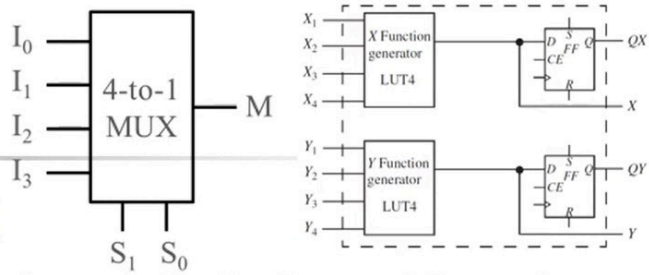

A 4-to-1 multiplexer




A building block of the FPGA



# <Ans.>

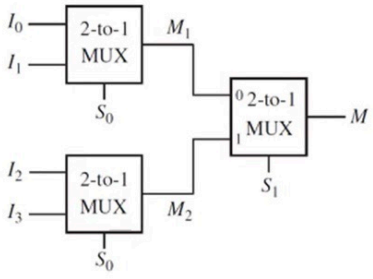



- For the 4-to-1 MUX:
  - Inputs: 6; multiplexer inputs $I_0$, $I_1$, $I_2$, and $I_3$ and multiplexer selects $S_1$ and $S_0$
  - Output equation:

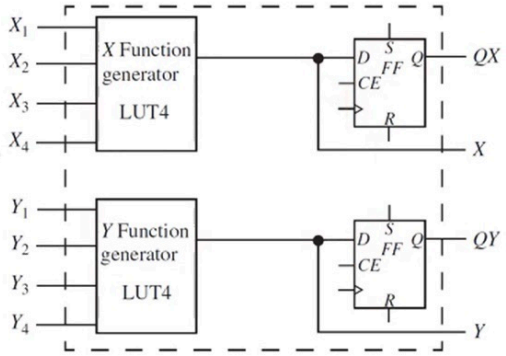
$$M = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$
$$= S_1'(S_0'I_0 + S_0I_1) + S_1(S_0'I_2 + S_0I_3)$$

  - Decompose into three 2-to-1 MUXs:

$$M_1 = S_0'I_0 + S_0I_1$$
$$M_2 = S_0'I_2 + S_0I_3$$
$$M = S_1'M_1 + S_1M_2$$



$$M_1 = S_0'I_0 + S_0I_1$$
$$M_2 = S_0'I_2 + S_0I_3$$
$$M = S_1'M_1 + S_1M_2$$




A building block of the FPGA

- **Two** logic blocks will be required to implement a 4-to-1 MUX.
  - The functions generated by the 1st logic block:

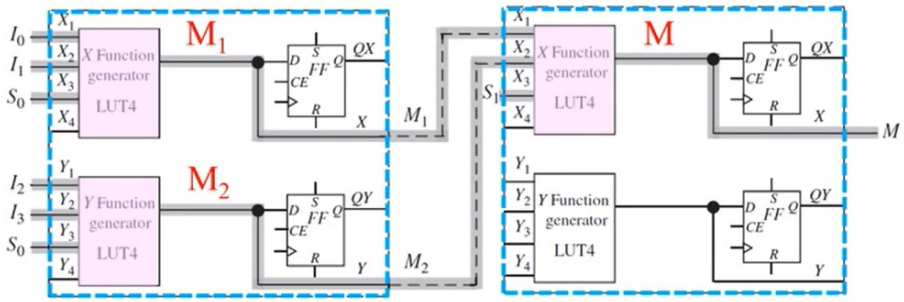
    $X = M_1 = S_0'I_0 + S_0I_1$

    $Y = M_2 = S_0'I_2 + S_0I_3$
  - Only half of the 2nd logic block is used:

    $X = M = S_1'M_1 + S_1M_2$
  - The flip-flops are unused in this design.



$$M_1 = S_0'I_0 + S_0I_1$$
$$M_2 = S_0'I_2 + S_0I_3$$
$$M = S_1'M_1 + S_1M_2$$




- The contents of the look-up tables:
  - Assumption: $X_1$ and $Y_1$ are the LSBs, and $X_4$ and $Y_4$ are the MSBs of the LUT addresses.
  - Truth table of LUT-M1: a 2-to-1 MUX

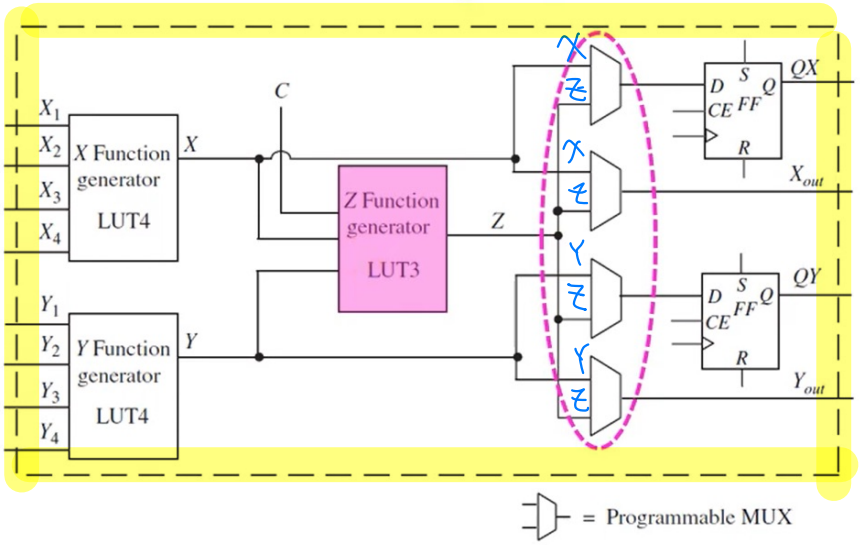
| Inputs | | | | Output |
|---|---|---|---|---|
| $X_4$ | $X_3$ ($S_0$) | $X_2$ ($I_1$) | $X_1$ ($I_0$) | $X$ |
| x | 0 | 0 | 0 | 0 |
| x | 0 | 0 | 1 | 1 |
| x | 0 | 1 | 0 | 0 |
| x | 0 | 1 | 1 | 1 |
| x | 1 | 0 | 0 | 0 |
| x | 1 | 0 | 1 | 0 |
| x | 1 | 1 | 0 | 1 |
| x | 1 | 1 | 1 | 1 |

  - The contents of LUT-M1:

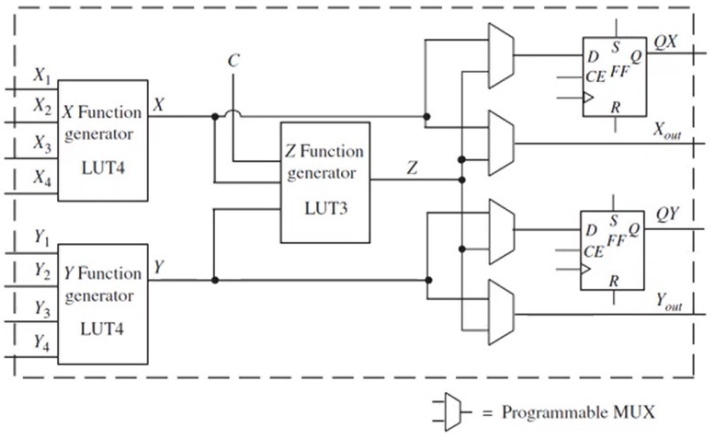
    LUT-M1– 0, 1, 0, 1, 0, 0, 1, 1,
    0, 1, 0, 1, 0, 0, 1, 1
  - LUT-M2, LUT-M: the same as LUT-M1

# Implementing Functions in FPGAs

- Some FPGAs provide two 4-variable function generators and a method to combine the output of the two function generators.
  - E.g.:






- The programmable logic block has ***nine*** logic inputs:

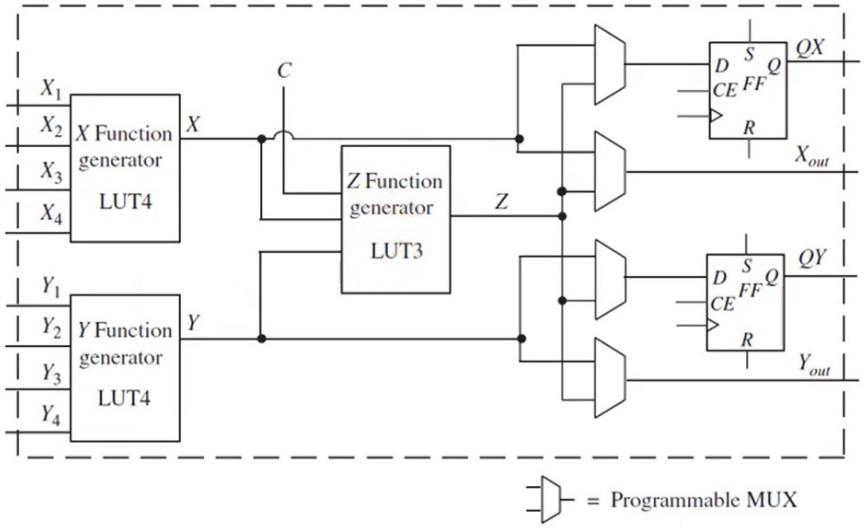
  $X_1, X_2, X_3, X_4, Y_1, Y_2, Y_3, Y_4$, and $C$

- It can generate ***two*** independent functions of ***four*** variables, $f_1(X_1, X_2, X_3, X_4)$ and $f_2(Y_1, Y_2, Y_3, Y_4)$, and a function $Z$ which depends on $f_1, f_2$, and $C$, $Z(f_1, f_2, C)$.
- It can generate any function of ***five*** variables in the form $Z = f_1(F_1, F_2, F_3, F_4) \cdot C' + f_2(F_1, F_2, F_3, F_4) \cdot C$, and can generate some functions of 6 to 9 variables.

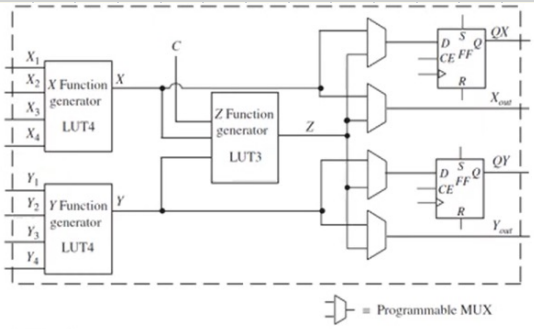
Ex. 4x1 MUX

# Example: 4-to-1 MUX

- Consider the implementation of a 4-to-1 MUX using a single logic block of this FPGA.




6-15




- X function generator (LUT4) generates the function: $F_1 = S_1'S_0'I_0 + S_1'S_0I_1$
- Y function generator (LUT4) generates the function: $F_2 = S_1S_0'I_2 + S_1S_0I_3$
- Z function generator (LUT3) performs an OR function of the $F_1$ and $F_2$ functions: $Z = F_1 + F_2$

* In this case, the *C* input is not required.

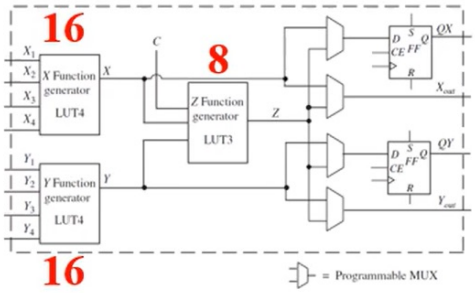
* Often, there are many ways to map the same design.

* It is very expensive to create multiplexers using LUTs.

J.J. Shann 6-18

# Resources in FPGAs

- SRAM cells:
  - 3-input function generator (LUT3): 8 SRAM cells
  - 4-input function generators (LUT4): 16 SRAM cells
- E.g.:
  - 40 memory cells

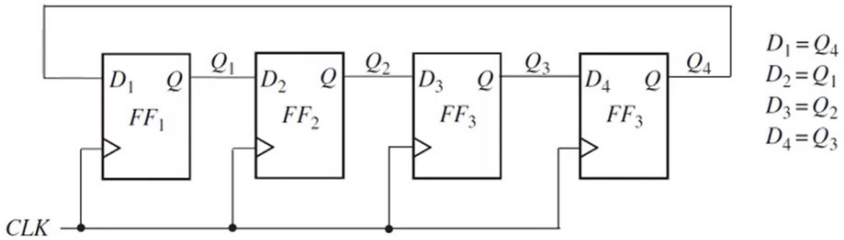



- When the ***programmable logic block*** of an FPGA is a large unit w/ ability to realize a fairly complex multivariable function, it is possible that a large part of each logic block may go unused.

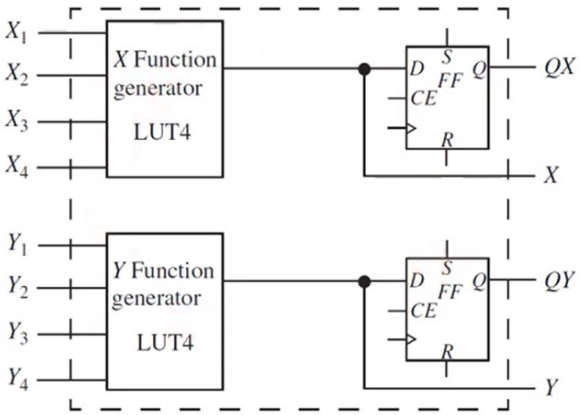


Ex. Shift Register

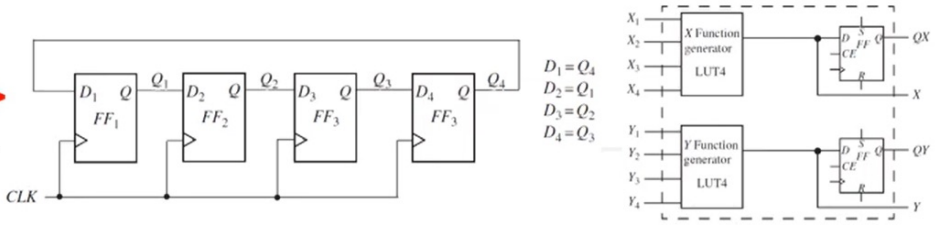
# Example

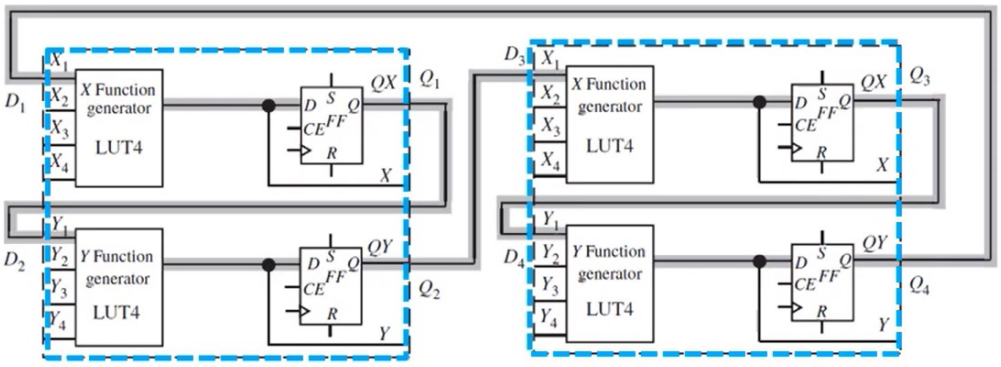



- Design a 4-bit ***circular shift register***, i.e., a ***ring counter***, in an FPGA whose building block is shown below.



# <Ans.>




* The 4-variable function generators are largely unused in this example.

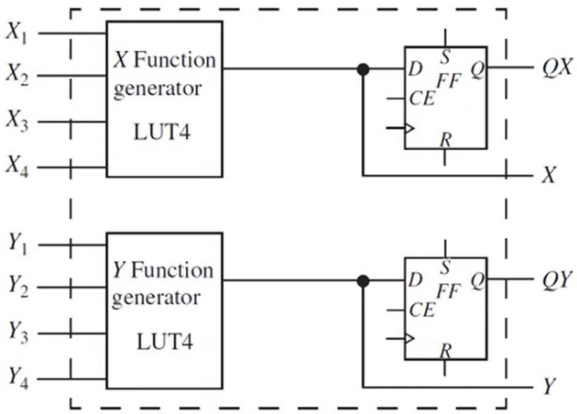
* Even if a function generator is used for a single variable function the rest of the function generator cannot be used for anything else.



Ex. 3×8 decoder

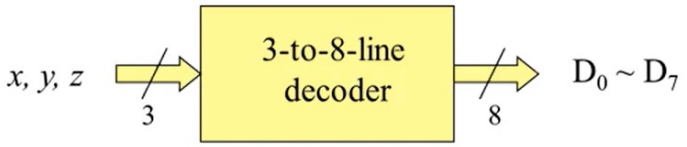
# Example: 3-to-8 Decoder

- How many programmable logic blocks similar to the one below will be required to create a 3-to-8 decoder?



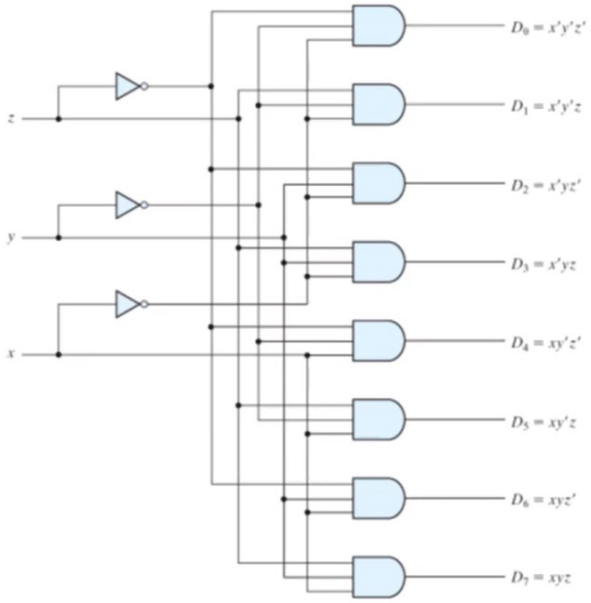
# (Review)

- 3-to-8 (line) decoder: active-HIGH outputs



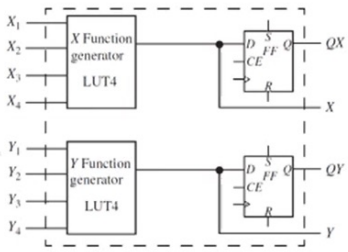

*Truth Table of a Three-to-Eight-Line Decoder*

| Inputs | | | Outputs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| x | y | z | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |






# <Ans.>




- Answer: Four.
  - A 3-to-8 decoder has three inputs and eight outputs.
  - Each output will need a 3-variable function generator.
  - Since what is available in the logic block is a 4-variable function generator, one will have to use one such function generator to create one output.

⇒ Eight function generators will be required to create a 3-to-8 decoder.

⇒ Four programmable logic blocks will be required to create a 3-to-8 decoder. (One logic block can generate two outputs.)

- different building blocks of FPGAs
  - Some FPGAs use muxes and gates as basic building blocks.
  - Some FPGAs (e.g. Xilinx) provide LUTs and muxes.
  - The mapping software looks at the resources available in the target technology (i.e. the specific FPGA that is used) and translates the design into the available building blocks.