# Lecture 20

# Designing with Field Programmable Gate Arrays (Chapter 6)

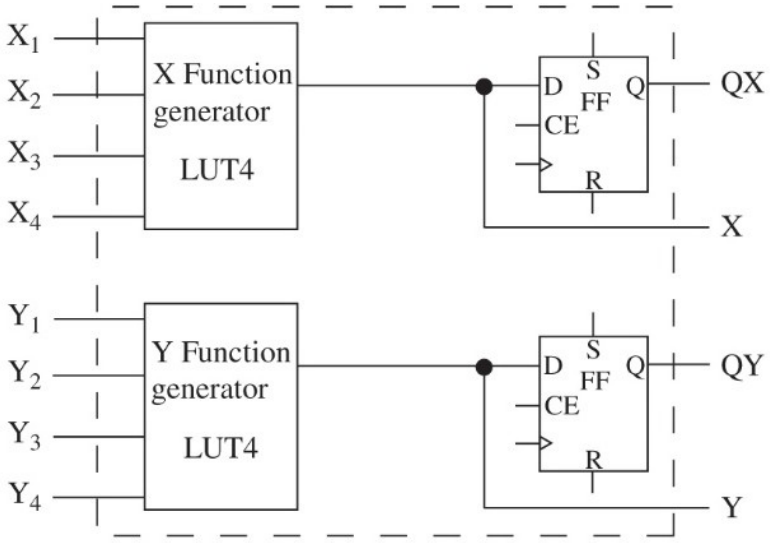
Khaza Anuarul Hoque
ECE 4250/7250

# Implementing Functions in FPGAs

- To understand issues associated with partitioning a design into an FPGA, some small components using FPGAs are designed in this section.

# Implementing Functions in FPGAs (continued)

- Design example: 4-to-1 multiplexer using look-up tables (LUTs) and flip-flops:

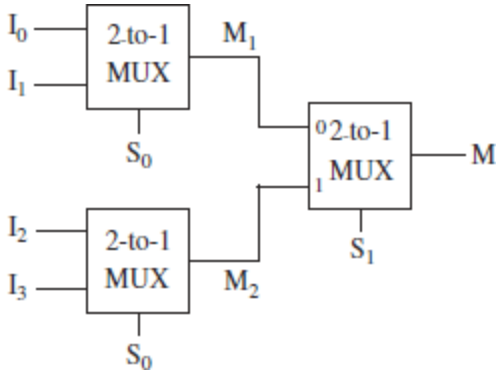


# Implementing Functions in FPGAs (continued)

- Design example (continued):
  - Contains two 4-variable function generators: X and Y, and 2 flip-flops.
  - Latched or unlatched forms of the generated functions can be brought to the output of the logic block. Latched outputs are *QX* and *QY*; combinational outputs are X and Y. With multiplexer inputs $I_0$, $I_1$, $I_2$, and $I_3$, and multiplexer selects $S_1$ and $S_0$, the output equation is:
    - $M = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$

# Implementing Functions in FPGAs (continued)

- Design example (continued):
  - A 4-to-1 multiplexer can be decomposed into three 2-to-1 multiplexers as illustrated:




  - $M_1 = S_0'I_0 + S_0I_1$
  - $M_2 = S_0'I_2 + S_0I_3$

# Implementing Functions in FPGAs (continued)

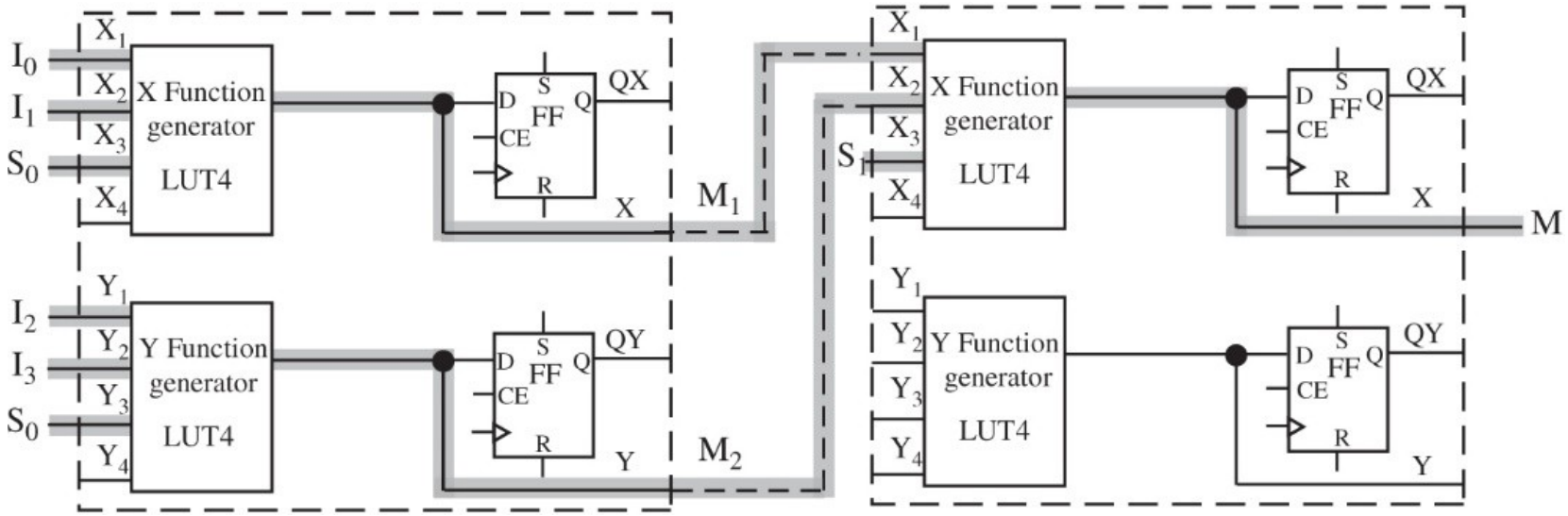
- Design example (continued):
  - A third 2-to-1 multiplexer must now be used to create the output of the 4-to-1 multiplexer:
    - $M = S_1'M_1 + S_1M_2$
  - Two logic blocks will be required to implement a 4-to-1 multiplexer using this type of logic block. The functions generated by the first logic block are:
    - $X = M_1 = S_0'I_0 + S_0I_1$
    - $Y = M_2 = S_0'I_2 + S_0I_3$

# Implementing Functions in FPGAs (continued)

- Design example (continued):
  - Only half of the second logic block is used. The X function generator creates the function:
    - $M = S_1'M_1 + S_1M_2$
  - The flip-flops are unused in this design.
  - Many modern FPGAs use an LUT4 as a basic building block. It can implement a function (1-bit) of any 4 variables. It takes 16 bits of SRAM to realize the LUT4 using the SRAM technology.

# Implementing Functions in FPGAs (continued)

- Design example (continued):
  - Path used by $M_1$ and $M_2$:



As illustrated in the figure, three look-up tables are used to implement functions M1, M2, and M. All of them are essentially 2-to-1 multiplexers. Assuming $X_1$ and $Y_1$ are the LSBs and $X_4$ and $Y_4$ are the MSBs of the LUT addresses, one can create the truth tables for each LUT as shown. When $S_0$ is 0, the output ($X$) equals $I_0$, and when $S_0$ is 1, the output equals $I_1$. Denote the three LUTs as LUT-M1, LUT-M2, and LUT-M.

| 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 MSB of each LUT is unused. The contents of the first 8 locations of the LUT should be duplicated for the next eight locations, since irrespective of the value of $X_4$, it can be expected to behave like a 2-to-1 multiplexer. Hence, the contents of LUT-M1 are the following:

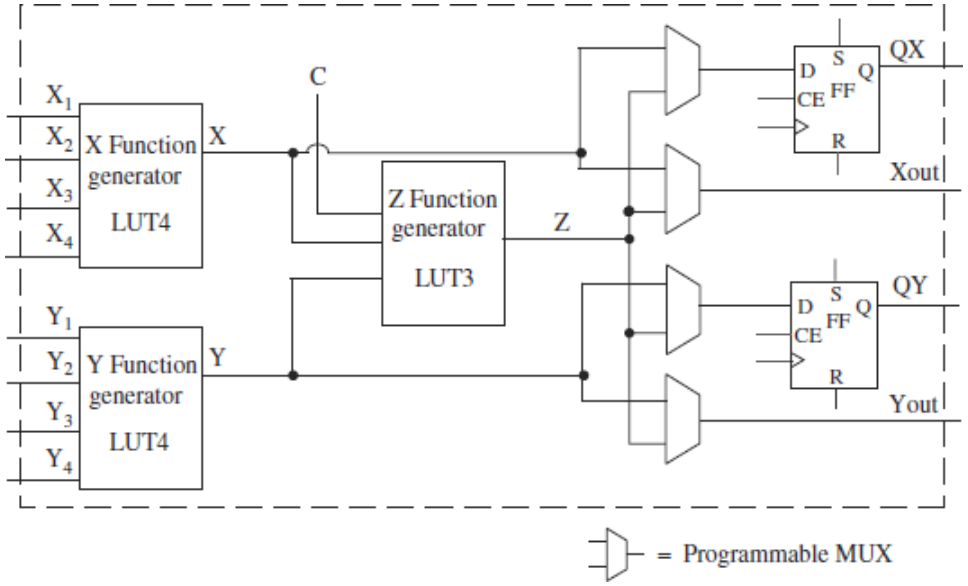
LUT-M1 – 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1

Since all three LUTs in Figure 6-2 are implementing 2-to-1 multiplexers, they have identical contents for the input connections shown. The contents of the second and third LUTs are the following:

LUT-M2 – 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1
LUT-M – 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1

# Implementing Functions in FPGAs (continued)

- Some FPGAs provide two 4-variable function generators and a method to combine the output of the two function generators. Consider the logic block shown below:

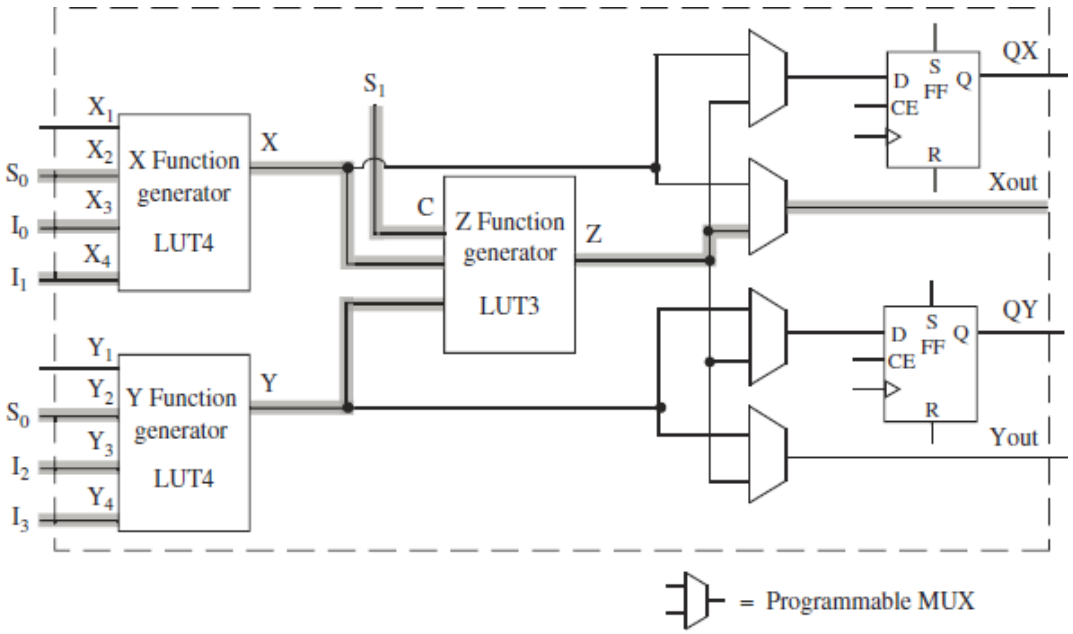


# Implementing Functions in FPGAs (continued)

- 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 $C$.
- It can generate any function of 5 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.

# Implementing Functions in FPGAs (continued)

- Using a single logic block from the preceding illustration, consider the implementation of a 4-to-1 MUX:



# Implementing Functions in FPGAs (continued)

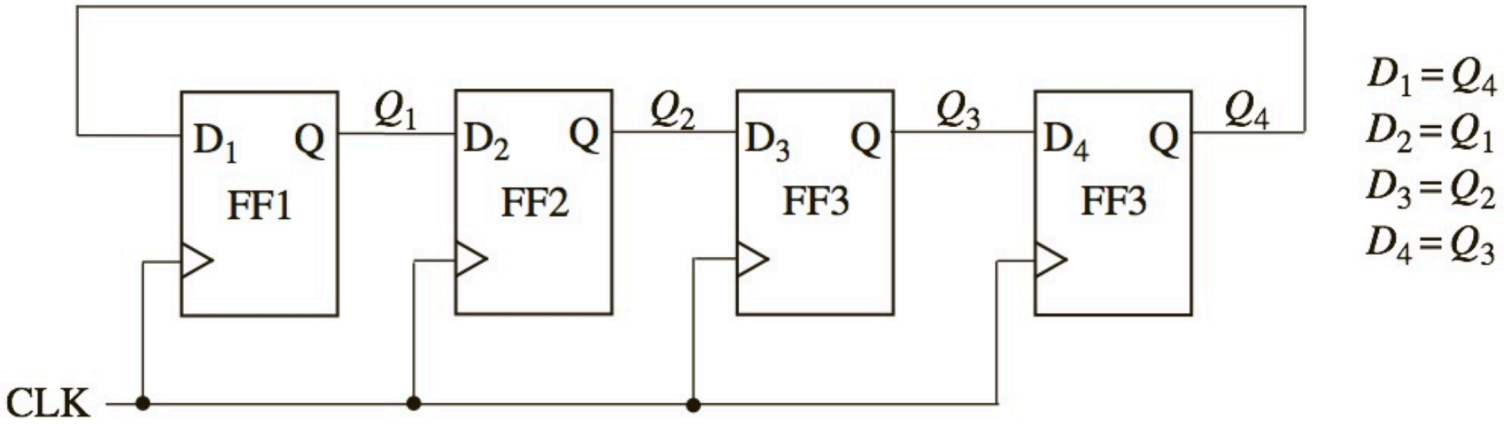
- 4-to-1 MUX (continued):
  - X function generator (LUT4): $M_1 = S_0'I_0 + S_0I_1$
  - Y function generator (LUT4): $M_2 = S_0'I_2 + S_0I_3$
  - Z function generator: $M = S_1'M_1 + S_1M_2$
  - Input *C* is used to feed in select signal $S_1$ for use in the *Z* function generator.
  - Needs no flip-flops or latches.

# Implementing Functions in FPGAs (continued)

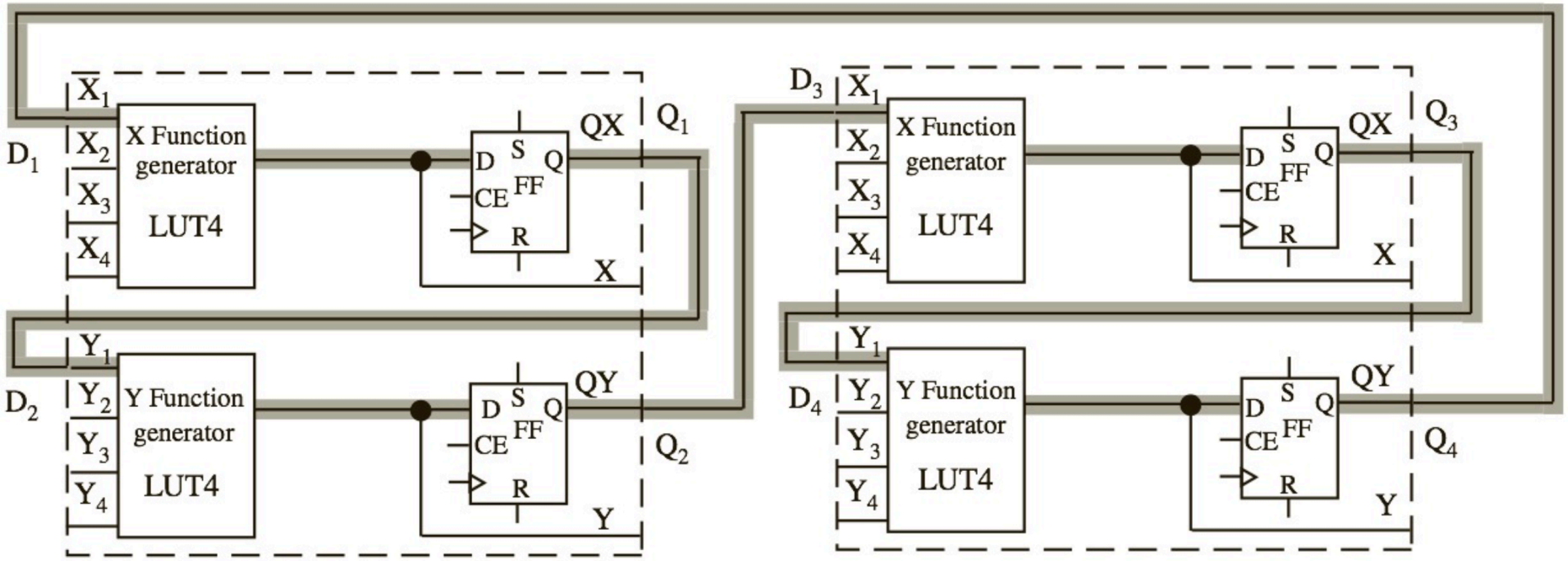
- There are many ways to map the same design. The MUX presented can be created even without using the C input:
  - The first two terms of the MUX's equation have 4 variables: $S_0$, $S_1$, $I_0$, and $I_1$.
  - The third and fourth terms have 4 variables: $S_0$, $S_1$, $I_2$, and $I_3$.
  - So, a 4-variable function generator can implement the first two terms, and another 4-variable function generator can implement the third and fourth terms.

# Implementing Functions in FPGAs (continued)

- Next, combine the outputs of the two 4- variable function generators using the Z function generator.
  - 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:
    $Z = F_1 + F_2$
- It is very expensive to create multiplexers using LUTs.


D1 Q
FF1
D2 Q
FF2
D3 Q
FF3
D4 Q
FF3
Q1
Q2
Q3
Q4
CLK
D1 = Q4
D2 = Q1
D3 = Q2
D4 = Q3

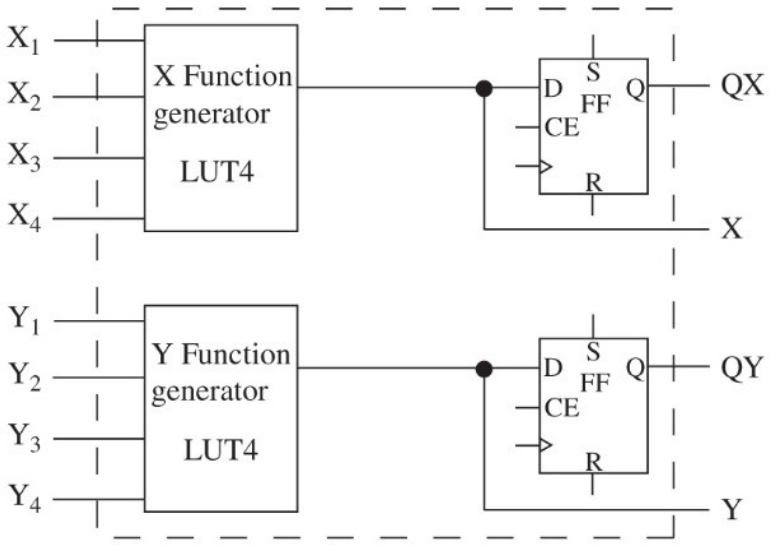
(a)


X1
X2
X3
X4
X Function generator
LUT4
D1
QX
Q1
S
D
Q
CE
FF
R
X
Y1
Y2
Y3
Y4
Y Function generator
LUT4
D2
QY
Q2
Y
D3
D4
Q3
Q4

(b)

# Implementing Functions in FPGAs (continued)

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



# Implementing Functions in FPGAs (continued)

- 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.
  - Thus, eight function generators will be required. One logic block can generate two outputs. Thus, four such programmable logic blocks will be required.

# Implementing Functions in FPGAs (continued)

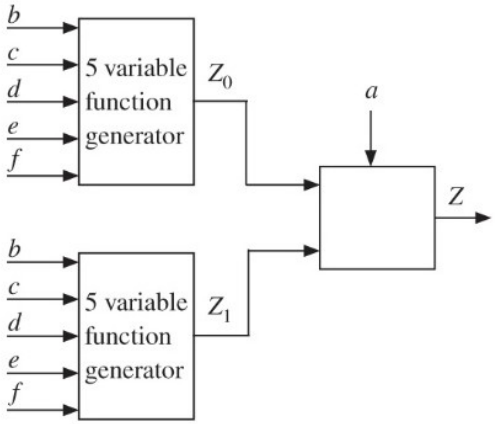
- Some FPGAs use multiplexers and gates as basic building blocks.
- Some FPGAs (e.g., the Xilinx Spartan) provide LUTs and multiplexers.
- The mapping software looks at the resources available in the target technology (i.e., the specific FPGA that is used) and translates the design into the available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

# Implementing Functions Using Shannon's Decomposition

- Shannon's Decomposition: used to decompose functions of large numbers of variables into functions of fewer variables.
- To demonstrate: use theorem for realizing any 6-variable function $Z(a, b, c, d, e, f)$.
  - First expand it:
    - $Z(a, b, c, d, e, f) = a' \cdot Z(0, b, c, d, e, f) + a \cdot Z(l, b, c, d, e, f) = a'Z_0 + aZ_1$
  - Verify that the equation is correct by first setting $a$ to 0 on both sides and then setting $a$ to 1 on both sides.

# Shannon's Decomposition (continued)

- Since the equation is true for both $a = 0$ and $a = 1$, it is always true. The equation leads directly to the circuit shown below:




- The circuit uses 2 cells to realize $Z_0$ and $Z_1$ and half of a 3rd cell to realize 3-variable function $Z = a'Z_0 + aZ_1$

# Shannon's Decomposition (continued)

- Example: consider the following:
  - $Z = abcd'ef' + a'b'c'def' + b'cde'f$
  - Setting $a = 0$ gives:
    - $Z_0 = 0 \cdot bcd'ef' + 1 \cdot b'c'def' + b'cde'f = b'c'def' +$ b'cde'f
  - Setting a = 1 gives:
    - $Z_1 = 1 \cdot bcd'ef' + 0 \cdot b'c'def' + b'cde'f = bcd'ef' + b'cde'f$

# Shannon's Decomposition (continued)

- Since $Z_0$ and $Z_1$ are 5-variable functions, each of them needs a 5-input LUT.
- If only 4-input LUTs are available, the 5-variable functions should be further decomposed into 4-variable functions. Apply the theorem twice, by expanding about *a* and then *b*. Or, it can be done in one step as follows:
  - $Z(a, b, c, d, e, f) = a'b' \cdot Z(0, 0, c, d, e, f) + a'b \cdot Z(0, 1, c, d, e, f) + ab' \cdot Z(1, 0, c, d, e, f) + ab \cdot Z(1, 1, c, d, e, f)$
  $= a'b' \cdot Y_0 + a'b \cdot Y_1 + ab' \cdot Y_2 + ab \cdot Y_3$

# Shannon's Decomposition (continued)

- Considering the decomposition of the function: $Z = abcd'ef' + a'b'c'def' + b'cde'f$

  apply the theorem around $a$ and $b$:
  - Substituting $a = b = 0$ gives

    $Y_0 = c'def' + cde'f$
  - Substituting $a = 0$, $b = 1$ gives $Y_1 = 0$
  - Substituting $a = 1$, $b = 0$ gives $Y_2 = cde'f$
  - Substituting $a = b = 1$ gives $Y_3 = cd'ef'$

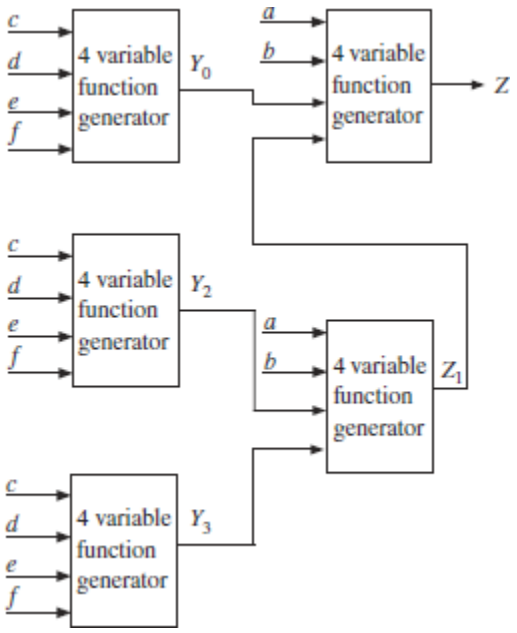
# Shannon's Decomposition (continued)

- One of the 4-variable functions obtained by decomposing is the null function, which results in a simpler function:
  - $Z = a'b' \cdot Y_0 + ab' \cdot Y_2 + ab \cdot Y_3$
  - Five 4-variable function generators will be sufficient to implement this function.

# Shannon's Decomposition (continued)

- Implementation of the function:

$Z = abcd'ef' + a'b'c'def' + b'cde'f$, using only 4-variable function generators:



# Shannon's Decomposition (continued)

- Shannon's decomposition allows one to decompose an *n*-variable function into 2 *n*-1 variable functions and multiplexers.
- It is very inefficient to realize multiplexers using LUTs. As the number of variables (*n*) increases, the number of look-up tables required to realize an *n*-variable function increases rapidly.

# Shannon's Decomposition (continued)

- The availability of multiplexers can greatly reduce the number of LUTs needed. Some FPGAs provide multiplexers in addition to LUT4s.
- The Xilinx Spartan FPGA is an example of an FPGA that provides multiplexers in addition to the general 4-variable LUTs. A logic unit in these FPGAs is called a **slice**.

# Shannon's Decomposition (continued)

Implement a seven-variable function using four-input LUTs and 2-to-1 multiplexers.

**Answer:**

Shannon's expansion can be used to obtain the following decompositions:

$$\text{7-variable function generator} = \text{two 6-variable function generators} + \text{a 2-to-1 mux} \ldots \quad \text{(i)}$$

$$\text{6-variable function generator} = \text{two 5-variable function generators} + \text{a 2-to-1 mux} \ldots \quad \text{(ii)}$$

$$\text{5 variable function generator} = \text{two 4-variable function generators} + \text{a 2-to-1 mux} \ldots \quad \text{(iii)}$$

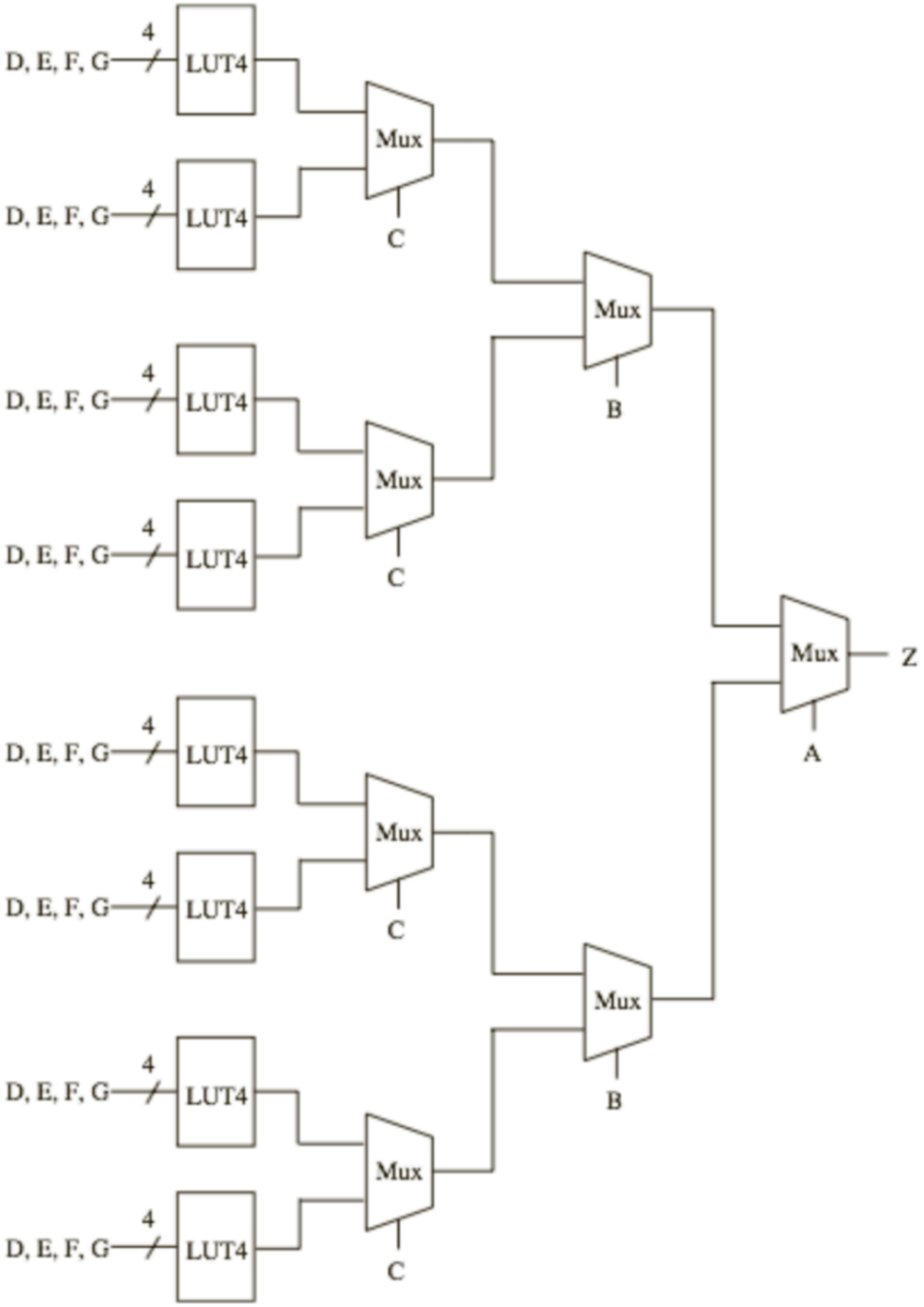
Substituting (iii) into (ii), we obtain

$$\text{6-variable function generator} = \text{four 4-variable function generators} + \text{three 2-to-1 muxes} \ldots \quad \text{(iv)}$$

Substituting (iv) into (i), we obtain

$$\text{7-variable function generator} = \text{eight 4-variable function generators} + \text{seven 2-to-1 muxes}$$

Thus a seven-variable function can be implemented as in Figure 6-8.


D, E, F, G
4
LUT4
Mux
C
B
A
Z