Use 4:1 Mux 74153 And Necessary Gate To Implement The Following Function: F = (0 To 5,7,8,12) =(10,11)

Use 4:1 Mux 74153 And Necessary Gate To Implement The Following Function: F = (0 To 5,7,8,12) =(10,11)

Introduction

Implementing combinational logic functions efficiently is a fundamental task in digital design. One common approach involves utilizing multiplexer (mux) devices to simplify circuitry, especially when the logic functions can be mapped to select lines and data inputs of the mux. The 74153 is a dual 4:1 multiplexer IC capable of selecting between four inputs based on two select lines, making it suitable for implementing multiple logic functions with minimal hardware.

In this article, we explore how to implement the specific function:

$$ F = \begin{cases}
1, & \text{for minterms } 0, 1, 2, 3, 4, 5, 7, 8, 12, 10, 11 \\
0, & \text{otherwise}
\end{cases} $$

This function is defined over a 4-bit input space, and the goal is to realize it using a 74153 4:1 multiplexer along with necessary logic gates.

Understanding the Function and Minterm Mapping

Input Variables and Minterms

Assuming the variables are A, B, C, D, representing the 4-bit input, the minterms are numbered from 0 to 15, corresponding to binary combinations:

Minterm	Binary (A B C D)	Decimal Value
0	0000	0
1	0001	1
2	0010	2

3	0011	3
4	0100	4
5	0101	5
7	0111	7
8	1000	8
10	1010	10
11	1011	11
12	1100	12

The function F outputs '1' for the minterms listed above and '0' elsewhere.

Truth Table and Minterm List

Constructing the truth table for all 16 minterms:

A	B	C	D	F
0	0	0	0	1	(m0)
0	0	0	1	1	(m1)
0	0	1	0	1	(m2)
0	0	1	1	1	(m3)
0	1	0	0	1	(m4)
0	1	0	1	1	(m5)
0	1	1	0	0	
0	1	1	1	1	(m7)
1	0	0	0	1	(m8)
1	0	1	0	0	
1	0	1	1	0	
1	0	1	1	0	
1	1	0	0	1	(m12)
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	0	

Implementation Strategy Using 74153 Multiplexer

The 74153 4:1 Multiplexer Overview

The 74153 is a dual 4:1 multiplexer, which has:

- Two select inputs: S0, S1

- Four data inputs: D0, D1, D2, D3
- Two enable inputs: G1, G2 (active low)
- Two outputs: Y and Ȳ (complement)

For our purpose, we will use one of the two multiplexers to implement the function, configuring the data inputs and select lines appropriately to produce F.

Mapping the Function to the MUX

The critical step is to determine how to assign the minterms to the multiplexer's data inputs and select lines.

Key points:

- The select lines of the mux will be driven by some combination of the input variables.
- The data inputs D0–D3 will be configured to output logic '1' or '0' depending on the minterms.

Design Approach

Step 1: Choose Select Lines

Decide which input variables will control the select lines (S0, S1). Typically, select lines are assigned based on the variables with the most significant impact or the simplest mapping.

Suppose we assign:

- S1 = A
- S0 = B

This means the mux selects among four groups based on A and B.

Step 2: Group Minterms Based on S1, S0

The minterms are grouped as follows:

S1 (A)	S0 (B)	Minterms	F=1?
0	0	0, 1, 2, 3, 4, 5	yes
0	1	7	yes
1	0	8, 10, 11, 12	yes
1	1	(none)	no

From the table, the minterms where F=1 correspond to:

- When A=0, B=0: minterms 0,1,2,3,4,5
- When A=0, B=1: minterm 7
- When A=1, B=0: minterms 8,10,11,12

Note: Minterm 7 (0111) falls into the group S1=0, S0=1; minterms 8,10,11,12 fall into S1=1, S0=0.

Remaining minterms (e.g., 13,14,15) are not in the function's minterms, so the output should be 0 for those.

Step 3: Define Data Inputs for Each Group

For each combination of select lines, define whether the data input D0–D3 should be '1' or '0'.

- D0 corresponds to (S1=0, S0=0): minterms 0–3
- D1 corresponds to (S1=0, S0=1): minterm 7
- D2 corresponds to (S1=1, S0=0): minterms 8,10,11,12
- D3 corresponds to (S1=1, S0=1): no minterms where F=1, so set to '0'

Thus:

Select Lines	Minterms	F=1?	Data Input
D0 (A=0, B=0)	0–3,4–5	Yes	1 (since minterms 0–5 are included)
D1 (A=0, B=1)	7	Yes	1
D2 (A=1, B=0)	8,10,11,12	Yes	1
D3 (A=1, B=1)	none	No	0

Implementing the Function with 74153

Step 1: Connect Select Lines

- S1 (A) connects to input A
- S0 (B) connects to input B

Step 2: Configure Data Inputs

- D0: connect to logic HIGH (1)

- D1: connect to logic HIGH (1)
- D2: connect to logic HIGH (1)
- D3: connect to logic LOW (0)

Step 3: Handling the Remaining Minterms

Since the function is high for the specified minterms, and the data inputs are set as above, the output of the mux will be high when the select lines select D0, D1, or D2, and low when D3 is selected.

Step 4: Dealing with Minterms Outside the Function

The remaining minterms (

Frequently Asked Questions

What is the main purpose of using a 4:1 MUX 74153 in implementing the function F = (0 to 5,7,8,12) = (10,11)?

The 4:1 MUX 74153 is used to select specific input combinations based on select lines, enabling the implementation of the specified function efficiently by routing the appropriate inputs to the output.

How do you configure the select lines of the 74153 MUX to realize the function F = (0 to 5,7,8,12) = (10,11)?

Configure the select lines to correspond to the binary representations of the input combinations that yield outputs 10 and 11, ensuring the MUX outputs these values for the specified minterms.

What role do necessary gates play in implementing the given function with the 74153 MUX?

Necessary gates, such as AND, OR, or NOT gates, are used to generate the correct logic levels or to combine signals to ensure the MUX outputs the desired function based on the input minterms.

How can the minterms (0 to 5,7,8,12) be mapped to the 4:1 MUX inputs?

The minterms are assigned to the MUX inputs based on their binary equivalent; for example, input 0 corresponds to minterm 0, input 1 to minterm 1, and so on, with specific inputs connected to logic high or low as needed.

Why is it necessary to use additional gates along with the 74153 MUX to implement the function F?

Additional gates are necessary to handle logic conditions not directly supported by the MUX, such as combining multiple minterms or implementing specific logic functions that require gating outside the MUX.

Can the given function F be implemented solely using the 74153 MUX without external gates?

In most cases, external gates are required to generate the correct input signals or to combine multiple minterms; the MUX alone may not be sufficient for the complete implementation of the function.

What is the significance of selecting minterms 10 and 11 for the function F in the design process?

Selecting minterms 10 and 11 specifies the output conditions for those particular input combinations, guiding how the MUX and gates are configured to produce the correct output for the function.

Additional Resources

Comprehensive Review of Using 4:1 MUX 74153 and Necessary Gates to Implement the Function F = Σ(0,1,2,3,4,5,7,8,12,10,11)

Introduction

In digital logic design, multiplexers (MUX) are versatile components that facilitate data selection among multiple inputs based on select lines. Among various types, the 74153 is a commonly used 4:1 multiplexer with dual 4-channel inputs, making it suitable for implementing complex functions efficiently. This review delves into the detailed process of implementing the Boolean function:

$$ F = \Sigma(0,1,2,3,4,5,7,8,12,10,11) $$

using the 74153 MUX and auxiliary logic gates. We will explore the function's minterm representation, mapping onto the multiplexer, selecting appropriate inputs, designing with necessary gates, and optimizing for hardware and logic efficiency.

Understanding the Function F and Its Minterm Representation

Deciphering the Function

The function \(F\) is a sum of minterms corresponding to the decimal numbers: 0, 1, 2, 3, 4, 5, 7, 8, 12, 10, 11. To understand how to implement this, it's essential to translate these into their binary forms and analyze the variables involved.

Assuming a 4-variable function with variables A, B, C, and D:

Minterm (Decimal)	Binary (A B C D)	Minterm Representation
0	0000	\(m_0\)
1	0001	\(m_1\)
2	0010	\(m_2\)
3	0011	\(m_3\)
4	0100	\(m_4\)
5	0101	\(m_5\)
7	0111	\(m_7\)
8	1000	\(m_8\)
10	1010	\(m_{10}\)
11	1011	\(m_{11}\)
12	1100	\(m_{12}\)

Note that minterms 6 and 9 are missing from the list, indicating these combinations do not contribute to the logic function output (F=0 for these inputs).

Summarizing the Minterm Function

The function can be expressed as a Boolean sum of minterms:

\[
F(A, B, C, D) = \sum_{i \in \{0,1,2,3,4,5,7,8,10,11,12\}} m_i
\]

This means \(F=1\) whenever the input combination matches any of these minterms; otherwise, \(F=0\).

Choosing the Implementation Strategy: Why Use a

74153 4:1 MUX?

Advantages of Using 4:1 MUX in Logic Implementation

The 74153 is a dual 4:1 multiplexer, capable of selecting one of four inputs based on a 2-bit select line. Its features include:

- Multiple Inputs and Select Lines: Two select lines (S0, S1) enable flexible input routing.
- High Speed and Reliability: Suitable for fast digital circuits.
- Simplicity in Implementation: Reduces the number of logic gates required for complex functions.
- Ease of Expansion: Can be combined with other gates to implement functions of multiple variables.

Using the 74153 simplifies the process of implementing functions that can be structured around selecting specific minterms or groups of minterms based on certain variable combinations.

Mapping the Function to a 4:1 MUX

The key to using a single 4:1 multiplexer lies in appropriately mapping the minterms onto the inputs of the MUX, with select lines corresponding to certain variables. Typically, the variables are partitioned into:

- Variables used as select lines: To determine which input to select.
- Variables used as data inputs: To set the logic levels at MUX inputs.

For example, choosing variables A and B as select lines (S1 and S0), the inputs \(I_0, I_1, I_2, I_3\) can be configured to represent the output for corresponding combinations of A and B, while C and D influence the input signals via logic gates.

Step-by-Step Implementation Process

Step 1: Variable Partitioning and Select Line Assignment

Given the minterm list, analyze which variables can serve as select lines:

- Assign variables A and B as select lines:
- S1 = A
- S0 = B

This partition simplifies the logic, as the minterms can be grouped based on A and B:

A	B	Minterms (with A, B fixed)	Minterm Numbers
0	0	0000, 0001, 0010, 0011, 0100, 0101	0,1,2,3,4,5
0	1	0111	7
1	0	1000, 1010, 1011	8,10,11
1	1	1100	12

Note the specific minterms in each group, which guides input configuration.

Step 2: Mapping Minterms onto the Multiplexer Inputs

For each combination of A and B (select lines), determine the values for the MUX inputs (\(I_0, I_1, I_2, I_3\)):

- When A=0, B=0 (S1=0, S0=0): covers minterms 0-5
- \(I_0\) should output 1 for minterms 0-5, 0 otherwise
- When A=0, B=1 (S1=0, S0=1): covers minterm 7
- \(I_1\) should output 1 if D=1 for minterm 7, else 0
- When A=1, B=0 (S1=1, S0=0): covers minterms 8, 10, 11
- \(I_2\) should output 1 for these minterms
- When A=1, B=1 (S1=1, S0=1): covers minterm 12
- \(I_3\) should output 1 for minterm 12

Step 3: Implementing MUX Inputs with Logic Gates

Each input line of the MUX can be set using logic gates based on the remaining variables (C, D) to produce the correct output for the minterms in each group.

- For \(I_0\) (A=0, B=0): Corresponds to minterms 0-5
- These include minterms with specific combinations of C and D.
- Use logic expressions to generate a high signal for minterms 0-5 and low otherwise.
- For example, \(I_0 = \text{OR}(\text{AND}(A', B', C', D'), \text{other minterm conditions})\)
- Similarly, configure \(I_1, I_2, I_3\) based on the minterm patterns.

Note: Since the minterms are scattered, it may be more efficient to generate each input signal using OR and AND gates that collectively cover all minterms mapped to that input.

Step 4: Implementing the Function Logic for Inputs

Design logic circuits for each MUX input:

- I0: Cover minterms 0-5
- Expression: \(I_0 = \text{OR of minterm conditions}\)
- For example:
\[

$$
I_0 = \text{OR}\left(
\text{AND}(A', B', C', D'), \quad
\text{AND}(A', B', C, D'), \quad
\text{AND}(A', B', C', D), \quad
\text{AND}(A', B', C, D)
\right)
$$
- I1: Cover minterm 7
- Expression:
$$
I_1 = \text{AND}(A', B, C, D)
$$
- I2: Cover minterms 8, 10, 11
- Expression:
$$
I_2 = \text{OR}\left(
\text{AND}(A, B', C', D'), \quad
\text{AND}(