Lecture 12: Designing System using Decoders

Prof. YingLi Tian
Oct. 26, 2016

Department of Electrical Engineering
The City College of New York
The City University of New York (CUNY)
A 2-to-4 Binary Decoder

A 2-to-4 Binary Decoder

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>0</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>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Combine Multiple Decoders to Form a Larger Decoder

- A 4-to-16 decoder using two 3-to-8 decoders.
- Inputs A, B, C are the inputs for each of the decoder, and D is used to select the appropriate one (D = 0 goes to the first decoder, D = 1 goes to the second decoder).
Use a Decoder to Enable other Decoders

- A 4-to-16 decoder using 5 2-to-4 decoders.
- Inputs c, d are the inputs for each of the decoder, and a, b are the inputs for the decoder that used to enable other decoders.
Implement Logic Functions using Decoders

Example 5.3 (p263 - 264): Implement the following functions using decoders with OR or NAND gates.

\[ f(a, b, c) = \sum m(0, 2, 3, 7) \]
\[ g(a, b, c) = \sum m(1, 4, 6, 7) \]
Implement Logic Functions using Decoders

Example 5.4 (p264 - 267): Implement the following functions using only 2-to-4 decoders and OR/NAND gates?

EN’ = 0, Active low

<table>
<thead>
<tr>
<th>EN’</th>
<th>a</th>
<th>b</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Implement Logic Functions using Decoders

Example 5.4 (p264 - 267): Implement the following functions using only 2-to-4 decoders and OR/NAND gates?

\[ f(a, b, c, d) = \sum m(0, 2, 3, 4, 7, 8, 10, 11, 15) \]
\[ g(a, b, c, d) = \sum m(0, 1, 2, 5, 8, 9, 10, 12, 13, 15) \]
\[ h(a, b, c, d) = \sum m(2, 4, 6, 8, 10, 12, 14, 15) \]
Implement Logic Functions using Decoders without Limits of the inputs of OR/NAND gates

Use 5 2-to-4 decoders:

\[
f(a, b, c, d) = \sum m(0, 2, 3, 4, 7, 8, 10, 11, 15)
\]
\[
g(a, b, c, d) = \sum m(0, 1, 2, 5, 8, 9, 10, 12, 13, 15)
\]
\[
h(a, b, c, d) = \sum m(2, 4, 6, 8, 10, 12, 14, 15)
\]

9 inputs for \( f \), 10 inputs for \( g \), and 8 inputs for \( h \).
Implementation: NANDs with less than 8 inputs.

\[ \begin{array}{c|cc|cc|cc}
\hline
& a & b & 00 & 01 & 11 & 10 \\
\hline
c & d & & 1 & 1 & 1 & 1 \\
00 & & 1 & 1 & & & \\
01 & & & & & & \\
11 & & & & & & \\
10 & & 1 & & & & \\
\hline
f & & & & & & \\
\end{array} \]

\[ \begin{array}{c|cc|cc|cc}
\hline
& a & b & 00 & 01 & 11 & 10 \\
\hline
c & d & & 1 & 1 & 1 & 1 \\
00 & & 1 & & & & \\
01 & & 1 & & & & \\
11 & & 1 & & & & \\
10 & & 1 & & & & \\
\hline
\begin{array}{c}
g \\
1, 5, 13, 9
\end{array} & & & & & & \\
\end{array} \]

\[ \begin{array}{c|cc|cc|cc}
\hline
& a & b & 00 & 01 & 11 & 10 \\
\hline
c & d & & 1 & & & & \\
00 & & 1 & & & & \\
01 & & 1 & & & & \\
11 & & & & & & \\
10 & & & & & & \\
\hline
\begin{array}{c}
h \\
2, 6, 14, 10
\end{array} & & & & & & \\
\end{array} \]
Use 4 2-to-4 decoders, NANDs with less than 8 inputs.
Implement Logic Functions using Decoders

Example 5.4 (p264 - 267): Implement the following functions using only 4 decoders shown below and 3 NAND gates (with as many inputs as necessary). Can you do it with only 4-input NANDs?

\[
\begin{align*}
  f(a, b, c, d) &= \sum m(0, 2, 3, 4, 7, 8, 10, 11, 15) \\
  g(a, b, c, d) &= \sum m(0, 1, 2, 5, 8, 9, 10, 12, 13, 15) \\
  h(a, b, c, d) &= \sum m(2, 4, 6, 8, 10, 12, 14, 15)
\end{align*}
\]
Implementation: NANDs with less than 4 inputs.

Step 1: draw K-Maps
Step 2: Simplify the functions, find minterm list.
Step 3: design the circuit to meet requirements (use NAND or OR gates)

\[
\begin{align*}
  f(a, b, c, d) &= \sum m(0, 2, 3, 4, 7, 8, 10, 11, 15) \\
  g(a, b, c, d) &= \sum m(0, 1, 2, 5, 8, 9, 10, 12, 13, 15) \\
  h(a, b, c, d) &= \sum m(2, 4, 6, 8, 10, 12, 14, 15)
\end{align*}
\]

NAND:
\[ f(a, b, c, d) = \sum m(0, 2, 3, 4, 7, 8, 10, 11, 15) = a'bc'd' + cd + b'd' \]

\[ g(a, b, c, d) = \sum m(0, 1, 2, 5, 8, 9, 10, 12, 13, 15) = abc'd' + abcd + c'd + b'd' \]

\[ h(a, b, c, d) = \sum m(2, 4, 6, 8, 10, 12, 14, 15) = abcd + cd' + bd' + ab'c'd' \]
Use NANDs with <= 4 inputs

1) \(a'c'd' + cd + b'd'\)
2) \(a'bc'd' + cd + b'd'\)

Figure in textbook
P267 is wrong!!

1) \(abc' + abd + c'd + b'd'\)
2) \(abc'd' + abcd + c'd + b'd'\)

1) \(abc + cd' + bd' + ac'd'\)
2) \(abcd + cd' + bd' + ab'c'd'\)
Announcement

- Read Chapter 5.2
- HW5 is out today, due on 11/02/2016
- Next class (Chapter 5.3, 5.4, 5.5) :
  - Encoders
  - Multiplexers
  - Demultiplexers
  - Three-State Gates