Decoders/Demultiplexors (DEMUX)

- Suppose we have $n$ components each of which has an associated line that can be used to enable/disable the component.

- We want to activate exactly one by causing an output of 1 on the associated line and 0 on all others.

- $\log n$ bits are needed to specify the index of the lines.

- We also assume that the DEMUX itself has an enable/diable line ($E$) such that when $E = 0$ all $n$ lines output a 0 and when $E = 1$ exactly one line outputs a 1.

- The line that outputs a 1 is given by decoding the binary integer specified by the $\log n$ input bits ($A_{\log n}, \ldots, A_0$).
Simplest case: 1-to-2 DEMUX

- $n = 2$ implies a 1-bit code and 2 output lines.

- Input $(E, A_0)$ chooses one or none of $(C_1, C_0)$ to output a 1.

- $A_0 = 0$ and $E = 1 \rightarrow C_0 = 1$ and $C_1 = 0$

- $A_0 = 1$ and $E = 1 \rightarrow C_0 = 0$ and $C_1 = 1$

- $A_0 = d$ and $E = 0 \rightarrow C_0 = 0$ and $C_1 = 0$

- This is simply the truth table for the multiple output circuit.
\[ \begin{array}{cc|cc}
E & A_0 & C_1 & C_0 \\
1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 \\
0 & d & 0 & 0 \\
\end{array} \]

- \( C_0 = \Sigma(2) = EA'_0 \)
- \( C_1 = \Sigma(3) = EA_0 \)
- \( C_1C_0 = 0 \) so MOPIs do not help.
2-to-4 DEMUX

- 2 inputs \((A_1, A_0)\) can encode 4 output indices.
- Setting \((E, A_1, A_0)\) determines \((C_3, C_2, C_1, C_0)\)

- \(C_0 = \Sigma(4) = EA'_1 A'_0\)
- \(C_1 = \Sigma(5) = EA'_1 A_0\)
- \(C_2 = \Sigma(6) = EA_1 A'_0\)
- \(C_3 = \Sigma(7) = EA_1 A_0\)

- \(C_i\) are distinct fundamental products so MOPIs do not help

- 4 AND gates required
$k$-to-$2^k$ DEMUX

- Assume we take the same AND-based approach
- $k+1$ input bits ($E, A_{k-1}, \ldots, A_0$)
- Assume $E$ is in most significant bit position
- $C_i = \Sigma (2^k + i)$
- Each $C_i$ has one distinct fundamental product associated with it so MOPIs do not help.
- $2^k$ AND gates each with $k+1$ input lines
- $E$ and all $A_i$ are fanned-out to all AND gates
- too costly
$k$-to-$2^k$ DEMUX

- A hierarchical implementation is possible

- Suppose you want to enable line $i$ corresponding to a 3-bit code word $(A_2, A_1, A_0)$, i.e., one of 8 output lines.

- Use bisection search idea to get an implementation in terms of 1-to-2 DEMUXs
• \((A_2, A_1, A_0) = (0, 1, 0)\)

• The initial interval of uncertainty is [0, 7], i.e., \(i\) could be any line.

• Question 1: Is \(i\) in the upper or lower half of the interval of uncertainty?

• Check \(A_2\):
  
  – \(A_2 = 0 \rightarrow 0 \leq i \leq 3\) (lower half)
  
  – \(A_2 = 1 \rightarrow 4 \leq i \leq 7\) (upper half)
  
  – We have new interval of uncertainty [0, 3]
• Question 2: Is $i$ in the upper or lower half of the interval of uncertainty, $[0, 3]$?

• Check $A_1$:
  - $A_1 = 0 \rightarrow 0 \leq i \leq 1$ (lower half)
  - $A_1 = 1 \rightarrow 2 \leq i \leq 3$ (upper half)
  - We have new interval of uncertainty $[2, 3]$
• Question 3: Is $i$ in the upper or lower half of the interval of uncertainty, $[2, 3]$?

• Check $A_0$:
  
  – $A_0 = 0 \rightarrow 2 \leq i \leq 2$ (lower half)
  
  – $A_0 = 1 \rightarrow 3 \leq i \leq 3$ (upper half)
  
  – We have determined line $i$, $i = 2$.

• $\log n$ checks, one for each bit in address of line.

• 2-to-4 DEMUXs or others can also be used
• Only 1 DEMUX enabled on each level.

• It enables exactly 1 DEMUX on the next level

• It disables exactly 1 DEMUX on the next level

• All disabled DEMUXs on a level disable exactly 2 DEMUX on the next level
Selector/Multiplexor (MUX)

- Suppose you have $n$ incoming lines and you want to choose one of them.

- The single output line will take on whatever value is present on the selected input line.

- $\log n$ bits are required to encode the index of the selected line.

- Example: a telephone with multiple lines – you select and converse on one at a time.

- several ways to implement them, including via a DEMUX and some AND gates (see text)
2-to-1 MUX

- 2 input data lines \((D_1, D_0)\)
- 1 bit needed to encode selection \((S_0)\)
- Output line \(Y = S'_0 D_0 + S_0 D_1\)
- Hierarchy can be built to get \(2^k\)-to-1 MUX
$2^k$-to-1 MUX

- Each input line index has a $k$-bit encoding
- The selected line is encoded $(S_{k-1}, \ldots, S_0)$
- We want to select the line whose encoding matches $(S_{k-1}, \ldots, S_0)$
- Compare each bit $S_i$ to the $i$-th bit of each line.
- Create a level of MUXs that perform these checks.
Hierarchical implementations of DEMUX and MUX are often used in a digital processor to connect a moderate number of resources.

A more powerful and scalable interconnection network for larger numbers of resources on a chip or board is a bus.

This requires devices in addition to switches.

A tristate driver must be used.

See the text for a discussion of a bus and other useful components (encoders, comparators, shifters).

We next discuss different ways of implementing Boolean functions using standard combinational components.