Implementation of 5 - 32 Bit Decoder using NAND Logic at 180 nm & 350 nm Technologies for High Speed Memory Applications

Anshika Sharma  
Department of E&C  
Moradabad Institute of Technology  
Moradabad, U.P., INDIA

Pankaj Bhardwaj  
Department of E&C  
Moradabad Institute of Technology  
Moradabad, U.P., INDIA

Alok Pandey  
Department of E, E & I  
Moradabad Institute of Technology  
Moradabad, U.P., INDIA

Abhinav Vishnoi  
Assistant Professor  
School of Electronics Engineering  
Lovely Professional University  
Phagwara, Punjab, INDIA  
abhnav.15929@lpu.co.in

Abstract

Discrete quantities of information can be represented in digital systems with binary codes. A binary code of n bits is capable of representing it up to 2n distinct elements of given information. A decoder is a combinational circuit that transforms binary information from n input lines to a maximum of 2n unique output ways. If the n-bit decoded information has unused or don’t care combinations, the decoder output will have less than 2n outputs.

1. INTRODUCTION

The major objective is to design all the necessary components required to form a 5 – 32 bit Decoder using NAND logic at 180 nm & 350 nm technologies. Most important aspect is to size the basic component of decoder (i.e., transistor) for a particular load. Discrete quantities of information can be represented in digital systems with binary codes. A binary code of n bits is capable of representing it up to 2n distinct elements of given information. A decoder is a combinational circuit that transforms binary information from n input lines to a maximum of 2n unique output ways. If the n-bit decoded information has unused or don’t care combinations, the decoder output will have less than 2n outputs.

Some Example: 2-4 decoder, 3-8 decoder, 5-32 decoder and so on. Figure shows a basic 2 to 4 Line Single Bit Decoder. It consists of two NOT Gate & 4 AND Gates.

The purpose of decoders is to generate 2n or less minterms of n input variables when applied. The name decoder is used in conjunction with some code converters such as BCD to Seven Segment Decoder. The output of the decoder may be further clarified from its input and output relationship.

A decoder with an enable input can function as a demultiplexer. Decoder/Demultiplexer logic circuits can be connected to form a large decoder circuit. A demultiplexer is a circuit that receives information on a single line and transmits the information on all 2n output lines. The selection of a specific output line is controlled by the bit values of n selection lines.

As the decoder results in 2n minterms of n input variables. So any Boolean function can be expressed as the sum of minterms canonical form, one can use decoder to generate the minterms and an external OR logic gate to form sum. Like this way, any combinational circuit with n inputs and 2n outputs can be implemented with an n to 2n decoder and OR logic gates.

The procedure for implementing any combinational circuit by means of decoder and OR logic gate requires that the Boolean Function of the circuit be expressed in sum of minterms. This form can be computed from truth table or by expanding the functions to their sum of minterms. A decoder is then chosen to generate all minterms of n input variables. The input for each OR gate is selected from decoder outputs according to minterm list in each function.

The method of designing decoder can be implemented for any combinational circuit. Its implementation must be compared with all other possible implementations to determine the best solution. In other cases, this decoder design method may provide

Table 1. The Truth Table of the above circuit is as follows:

<table>
<thead>
<tr>
<th>A1</th>
<th>A0</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 1. Basic 2 to 4 Line Single Bit Decoder
the best implementation for e.g. if the circuit has many output lines and if each output function is expressed with a small number of minterms.

II. DESIGN TECHNIQUES

The design is usually done with specific objectives depending on the application for which it is designed. This requires trade-offs to be made by the designer between various factors such as speed of operation, area of the design and hence cost involved, power dissipated etc. Mainly the optimizations made in design can be classified into three stages To achieve minimal power dissipation, Maximum operating speed, Minimal silicon area.

These three techniques are described below:

2.1 Low Power Optimization

Capacitance is the major contributing factor in power dissipation. So a library optimized for low power tries to reduce the capacitance as much as possible. This is done by using minimum size devices as much as possible. However the design should meet the noise margin criteria. Any noise from the power supply should not affect the logic levels. The logic thresholds are checked for all devices against temperature, power supply and process variations. Adjustments are made in the transistor sizes if the noise margin requirements are not met and this is iteratively done until the noise margin requirements are met. The output rise and fall times are limited by limiting the maximum capacitance that a gate can drive.

2.2 High Speed Optimization

High speed optimization aims to achieve minimal gate delays without compromising area. Devices can be sized up to increase the speed as more drive current is provided to charge and discharge the transistors. Usually high speed optimization is done by calculating the cell height for minimum sized devices. By minimum sized devices we mean the NMOS devices at minimum allowable size and the PMOS devices at 1.5 to 2 times the minimum size. This is done to ensure similar rise and fall characteristics as the PMOS device is weaker than the NMOS due to smaller mobility for holes. Then this cell height is made to increase nominally to 25% over this minimum and the delay characteristics are noted. Further increase in the height is only done if the increase in speed exceeds the square of the height increase. Iterative SPICE simulations are done to fit the devices accordingly in this increased height in order to obtain minimum gate delays. Finally, a static noise margin check is done to ensure that the noise margin requirements are met.

2.3 Minimal Area Optimization

Minimal area optimization is designed so as to get the fastest possible cells in the minimum area. The minimum standard cell height is fixed and the cells are sized to fill this height. This is subject to further checks to ensure that the noise margins and the gate output rise and fall times are satisfied. If not, iterative SPICE simulations are done to determine the increased sizes of the NMOS and PMOS devices. The library that was designed was optimized mainly for area. So most circuits were designed with the NMOS and PMOS set at the minimum widths that maintain correct functionality. The analog cells had specific design requirements, so minimum area was not the main design objective in those cases.

In the above figure sensing the spectrum is mainly used to identify the presence of user at power band in each level. At all levels it keep dividing in to low pass as well as the high pass filters. Here we are applying the leach algorithm for various thresholds for the obtained energy level. By using this we can find exactly the availability of primary user and can easily detect the spectrum holes. For the analysis of signal spectrum one of the good tool other than the FT is CWT. Because in FT it presumes the signal to be periodic and stationary.

2.4 Design Issues

While designing any circuit designer have to consider few parameters, necessary for the proper working of that circuit, these parameters are known as design issues.

Some of them are as follows:

- Delay
- Number of inputs
- Number of stages
- Size of transistor
- Load capacitance

2.5 H/w & S/w Require for Implementation

For the implementation of the schematics we are using Mentor Graphics IC Flow tool.

III. LOGICAL EFFORT CALCULATIONS FOR DELAY

We use Logical effort calculations for delay estimation. The method of logical effort is an easy way to estimate the delay in an MOS circuit. Logical efforts estimates the delay in logic circuits and selects the fastest logic circuit.

Delay in a circuit depends on:

- Load on the logic circuit.
- Number of logic gates in the logic circuit

3.1 Single Stage Logic Network

Delay can be divided into two parts:

Parasitic delay (P): Parasitic delay of a gate is the delay when it drives zero load. It is a fixed delay for a circuit. Parasitic delay is calculated by using the following table.
Table 2. Standard Value of Parasitic Delay's[1]

<table>
<thead>
<tr>
<th>Gate type</th>
<th>Parasitic delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>inverter</td>
<td>( p_{inv} )</td>
</tr>
<tr>
<td>( n )-input NAND</td>
<td>( n p_{inv} )</td>
</tr>
<tr>
<td>( n )-input NOR</td>
<td>( n p_{inv} )</td>
</tr>
<tr>
<td>( n )-input multiplexer</td>
<td>( 2 n p_{inv} )</td>
</tr>
<tr>
<td>XOR, XNOR</td>
<td>( 4 p_{inv} )</td>
</tr>
</tbody>
</table>

Effort Delay \((f)\): It is also known as the Stage delay [2]. It is proportional to the load on the gate’s output. It can be further divide in to two parts as follows:

Electrical Effort \((h)\): Electrical effort basically characterizes the load. It is also known as “Fan out” of the circuit and given by –

\[
h = \ln \left( \frac{C_L}{C_g} \right)
\]

\(C_L\) – Load Capacitance at the output of the gate

\(C_g\) – Input Capacitance at the gate

Logical Effort \((g)\): “Logical effort tells us that how much more input capacitance we require at the gate for given input to deliver the same output current as an inverter”. Logical Effort is calculated by using the following table :

Table 3. Standard Value of Logical Effort[1]

<table>
<thead>
<tr>
<th>Gate type</th>
<th>Numbers of inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>inverter</td>
<td>1</td>
</tr>
<tr>
<td>NAND</td>
<td>( \frac{3}{2} )</td>
</tr>
<tr>
<td>NOR</td>
<td>( \frac{3}{2} )</td>
</tr>
<tr>
<td>multiplexer</td>
<td>( \frac{3}{2} )</td>
</tr>
<tr>
<td>XOR (parity)</td>
<td>( \frac{n+2}{3} )</td>
</tr>
</tbody>
</table>

So the total delay of single stage network is given by

\[
D = f + P
\]

\(f = gh\)

So Delay \(D = P + gh\)

Multi Stage Network

For multi stage network Logical effort along the path in the network compounds by multiplying the logical effort of all the gates along the path so Path Logical Effort \((G)\) is given by –

\[
G = D g_i
\]

Path Electrical Effort \(H = C_L / C_g\)

\(C_L\): Capacitance that loads the last logic gate in path

\(C_g\): Capacitance of first gate in the path

When fan-out occurs at the output node, some of the available current is directed along the path we are analyzing and some directed off path, so we add a term Branching Effort [2] for the branching between stages and is given by –

Branching effort \((b)\)

\[
b = \frac{\text{Con-path+ Coff-path}}{\text{Con-path}}
\]

Con-path = Load Capacitance along the path we are analyzing

Coff-path = Capacitance of the connection that lead off path

Path Branching effort \((B)\) is given by –

\[
B = D b_i
\]

So the total Effort Delay \((F)\) is given by –

\[
F = GBH
\]

Path Parasitic Effort \((P)\) is given by –

\[
P = \hat{O} Pi
\]

Hence total Delay can be given as

\[
D = F + P
\]

\[
D = GBH + P
\]

Estimate the minimum delay \(D = NF^{iN} + \hat{O} \) using values of parasitic delay [1] obtained. If you are comparing different architectural approaches to a design problem, you may choose to stop the analysis here.

IV. IMPLEMENTATIONS OF DECODERS

The major objective is to design all the necessary components required to form a 5 – 32 bit Decoder using NAND logic at 180 nm & 350 nm technologies. Objective is to implement the following schematics:

1. NAND Gate (2 input).
2. NAND Gate (3 input).
3. 2—4 NAND Decoder based on 2 input NAND Gate.
4. 3—8 NAND Decoder based on 3 input NAND Gate.
5. 5—32 NAND Decoder based on 2 – 4 & 3 – 8 NAND Decoder.

4.1. NAND Logic Based Design

As the name implies the NAND logic based design, the circuits are implemented using NAND gate. To implement 2-4 decoder using NAND logic, a 2 input NAND gate is required. Similarly to implement 3-8 decoder using NAND logic, a 3 input NAND gate is required. The Schematic of NAND gate with the truth table and simulation results are shown in the figure as follows:

(a) NAND Gate (2 Input)

The Schematic of 2 input NAND gate with the truth table and simulation results are shown in the figure as follows:

Table 4. Truth Table of 2 input NAND gate

<table>
<thead>
<tr>
<th>S.No.</th>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A 0</td>
<td>B 0</td>
</tr>
<tr>
<td>2</td>
<td>A 1</td>
<td>B 1</td>
</tr>
<tr>
<td>3</td>
<td>A 0</td>
<td>B 1</td>
</tr>
<tr>
<td>4</td>
<td>A 1</td>
<td>B 1</td>
</tr>
</tbody>
</table>
Waveform (Transient Result):

Waveform shown in figure verifies the truth table shown in the table, as when all the inputs are high the output goes low otherwise it remains high.

(b) NAND Gate (3 Input)

The Schematic of 3 input NAND gate with he truth table and simulation results are shown in the figure as follows:

Table 5. Truth Table of 3 input NAND gate

<table>
<thead>
<tr>
<th>S. No.</th>
<th>INPUTS</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Waveform shown in figure verifies the truth table shown in the table, as when all the inputs are high the output goes low otherwise it remains high.

4.1.1. 2-4 Decoder Based on NAND Logic

2-4 decoder can be implemented by using NAND logic. Designer can calculate the size of the transistor of a NAND gate to drive a particular load. For example – Suppose we want to drive a 15 \( \mu \text{F} \) load from a 2-4 decoder, so the size of transistor to drive the particular load can be calculated [1, 4] as follows:

(a) At 180 nm Technology:

Known Parameters:

- Load Capacitance \( C_L = 15 \ \mu \text{F} \)
- No. of Stages \( N = 1 \)
- Channel Length \( L = 180 \ \text{nm} \)
- Oxide Thickness \( T_{\text{ox}} = 4.1 \ \text{nm} \)
- No. of stages \( N = \ln \left( \frac{C_L}{C_g'} \right) \)
- \( C_g' = 5.5181 \ \mu \text{F} \)

We are having 2 i/p gate at this stage so total capacitance at the gate:

\[
C_g = 5.5181 \times \frac{4}{3} = 7.357 \ \mu \text{F}
\]

Hence we can calculate the size of transistor.

\[
W = 4.769 \ \mu \text{m}
\]

(b) At 350 nm Technology:

Known Parameters:

- Load Capacitance \( C_L = 15 \ \mu \text{F} \)
- No. of Stages \( N = 1 \)
- Channel Length \( L = 350 \ \text{nm} \)
- Oxide Thickness \( T_{\text{ox}} = 7.8 \ \text{nm} \)
- No. of stages \( N = \ln \left( \frac{C_L}{C_g'} \right) \)
- \( C_g' = 5.5181 \ \mu \text{F} \)

We are having 2 i/p gate at this stage so total capacitance at the gate:

\[
C_g = 5.5181 \times \frac{4}{3} = 7.357 \ \mu \text{F}
\]

Hence we can calculate the size of transistor.

\[
W = 4.6665 \ \mu \text{m}
\]
So now the size of transistor for each gate to be used in 2-4 decoder is known, so a 2-4 decoder can now easily implemented. Truth table and schematics of 2-4 decoder based on NAND logic is shown in figure.

Table 6. Truth Table of 2-4 Decoder based on NAND logic

<table>
<thead>
<tr>
<th>INPUTS (A B)</th>
<th>OUTPUTS (Y4 Y3 Y2 Y1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>1 1 1 0</td>
</tr>
<tr>
<td>0 1</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>1 0</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td>1 1</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>

4.1.2. 3-8 Decoder based on NAND Logic

3-8 decoder can be implemented by using NAND logic. Designer can calculate the size of the transistor of a NAND gate to drive a particular load. For example – Suppose we want to drive a 15 fF load from a 3-8 decoder, so the size of transistor to drive the particular load can be calculated[1][4] as follows:

(a) At 180 nm Technology:

Known Parameters:
- Load Capacitance C_L = 15 fF
- No. of Stages N = 1
- Channel Length L = 180 nm
- Oxide Thickness T_ox = 4.1 nm
- No. of stages N = ln (C_L / C_g')
- C_g' = 5.5181 fF

We are having 3 i/p gate at this stage so total capacitance at the gate:
- C_g = 5.5181 X (5/3) = 9.1969 fF

Hence we can calculate the size of transistor:
- W = 5.962 μm

(b) At 350 nm Technology:

Known Parameters:
- Load Capacitance C_L = 15 fF
- No. of Stages N = 1
- Channel Length L = 350 nm
- Oxide Thickness T_ox = 7.8 nm
- No. of stages N = ln (C_L / C_g')
- C_g' = 5.5181 fF

We are having 3 i/p gate at this stage so total capacitance at the gate:
- C_g = 5.5181 X (5/3) = 9.1969 fF

Hence we can calculate the size of transistor:
- W = 5.832 μm

So now the size of transistor for each gate to be used in 3-8 decoder is known, so a 3-8 decoder can now easily implemented. Truth table and schematics of 3-8 decoder based on NAND logic is shown in figure:

Table 7. Truth Table of 3-8 Decoder based on NAND logic

<table>
<thead>
<tr>
<th>S.No.</th>
<th>INPUTS (A B C)</th>
<th>OUTPUT (Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0 1 0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0 1 1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1 1 0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1 1 1</td>
<td>1</td>
</tr>
</tbody>
</table>
Waveform shown in figure verifies the truth table shown in the table, as:
- When all the inputs are the output goes low the Y1 output line is selected.
- When all the inputs are the output goes high the Y8 output line is selected.

4.1.3. 5-32 Decoder using 2-4 & 3-8 Decoder based on NAND logic

5-32 decoder can be implemented by using 2-4 & 3-8 Decoder based on NAND logic. Designer can calculate the size of the transistor of a NAND gate to drive a particular load. For example – Suppose we want to drive a 15 fF load from a 5-32 decoder, so the size of transistor to drive the particular load can be calculated as follows:

(a) At 180 nm Technology:

Known Parameters:
- Load Capacitance $C_L = 15$ fF
- No. of Stages $N = 3$
- Channel Length $L = 180$ nm
- Oxide Thickness $T_{ox} = 4.1$ nm

1. Calculation for Last Stage

No. of stages $N = \ln \left( \frac{C_L}{C_g'} \right)$

$C_g' = 0.7468$ fF

We are having 2 i/p gate at this stage so total capacitance at the gate:

$C_g = 0.7468 \times (4/3) = 0.9957$ fF

Hence we can calculate the size of transistor:

$W = 0.6455$ µm

2. For 3-8 Decoder Stage

Total Load Capacitance for this stage

$C_L = 15 + 0.9957 = 15.9957$ fF

No. of stages $N = \ln \left( \frac{C_L}{C_g'} \right)$

$C_g' = 0.79637$ fF

We have three input gate at this stage so

$C_g = 0.79637 \times (5/3) = 1.3272$ fF

Hence we can calculate the size of transistor

$W = 0.8604$ µm

3. For 2-4 Decoder Stage

Total Load Capacitance for this stage

$C_L = 15 + 0.9957 = 15.9957$ fF

No. of stages $N = \ln \left( \frac{C_L}{C_g'} \right)$

$C_g' = 0.79637$ fF

We have three input gate at this stage so
\[ C_g = 0.79637 \times (4/3) = 1.06182 \text{fF} \]

Hence we can calculate the size of transistor:

\[ C_g = \text{COX} \ WL \]
\[ W = C_g \ \text{TOX} / \text{COX} \ L \]
\[ W = 0.6883 \mu \text{m} \]

(b) At 350 nm Technology:

Known Parameters:

Load Capacitance \( C_L = 15 \text{ fF} \)
No. of Stages \( N = 3 \)
Channel Length \( L = 350 \text{ nm} \)
Oxide Thickness \( T_{\text{ox}} = 7.8 \text{ nm} \)

1. Calculation for Last Stage

No. of stages \( N = \ln \left( C_L / C_g' \right) \)
\[ C_g' = 0.7468 \text{ fF} \]

We are having 2 i/p gate at this stage so total capacitance at the gate:
\[ C_g = 0.7468 \times (4/3) = 0.9957 \text{fF} \]
Hence we can calculate the size of transistor:
\[ W = 0.6315 \ \mu \text{m} \]

2. For 3-8 Decoder Stage

Total Load Capacitance for this stage
\[ C_L = 15 + 0.9957 = 15.9957 \text{ fF} \]
No. of stages \( N = \ln \left( C_L / C_g' \right) \)
\[ C_g' = 0.79637 \text{ fF} \]

We have three input gate at this stage so
\[ C_g = 0.79637 \times (5/3) = 1.3272 \text{ fF} \]
Hence we can calculate the size of transistor:
\[ W = 0.8418 \ \mu \text{m} \]

3. For 2-4 Decoder Stage

Total Load Capacitance for this stage
\[ C_L = 15 + 0.9957 = 15.9957 \text{ fF} \]
No. of stages \( N = \ln \left( C_L / C_g' \right) \)
\[ C_g' = 0.79637 \text{ fF} \]

We have three input gate at this stage so
\[ C_g = 0.79637 \times (4/3) = 1.06182 \text{ fF} \]
Hence we can calculate the size of transistor:
\[ W = 0.6735 \mu \text{m} \]

So now we have the size of transistor for each stage, so we can design the 5-32 decoder easily.

Truth table and schematics of 5-32 decoder based on NAND logic is shown in figure.
Waveform (Transient Result):

![Waveform Image]

**Fig. 11.** Output Waveform of 5-32 Decoder based on NAND logic

Waveform shown in figure verifies the truth table shown in the table, as:

- When both the inputs are the output goes low the Y1 output line is selected.
- When both the inputs are the output goes high the Y32 output line is selected.

V. CONCLUSION

In this paper a 5-32 decoder and all the necessary components to implement 5-32 decoder like 2-4 decoder, 3-8 decoder. Nand gate and Nor gate etc. are designed by using NAND logic. These implementations are designed on two technologies TSMC 180 nm and 350 nm. Comparative results are taken out by doing different analysis like temperature, load, frequency and delay analysis etc.

Power dissipation variations with respect to temperature, frequency and load are calculated for the decoders implemented using NAND logic.

REFERENCES


