Lab 03: I/O and POSTNET

Arturo Salinas-Aguayo
CSE 2301: Principles and Practice of Digital Logic Design
Dr. Mohammad Khan, Section 003L-1248
Electrical and Computer Engineering Department

College of Engineering, University of Connecticut
Coded in LaTeX

Theory

The 7405 Chip DIP

The **7405 DIP** *(Dual-Inline-Package)* from Texas Instruments is a different kind of inverter called an *Open Collector*, compared to the **7404 DIP** by the same manufacturer which is a standard inverter with something called a *Push-Pull Output Stage.* Our application requires driving a load, in this case some Light Emitting Diodes. These LEDs require higher current draw which a **7404** IC cannot handle directly. The **7405** provides the ability to drive an external load by pulling the output signal to ground without having to supply or drive the current from the IC itself.

The Bipolar Junction Transistor

In order to properly explain this choice, the *emitter, base,* and *collector* of a Bipolar Junction Transistor must be properly defined.

Example 1 ***The Bipolar Junction Transistor Abstraction***
Emitter A *heavily* doped, medium-sized layer designated to inject or emit electrons.
Base A thin layer of *medium* doped material designed for electron transmission.
Collector A wide layer of *lightly* doped material designed to *collect* electrons.
Texas Instruments utilizes NPN Transistors in their schematics such as this:

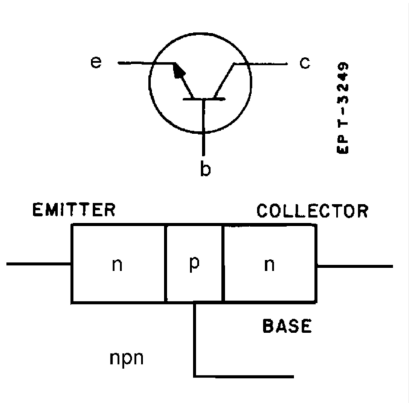


Figure 1: The NPN Transistor

Why the 7405?

All of this hand waving still doesn't answer the question as the difference hasn't been hammered home quite yet.

Example 2 shows the schematics from the official Texas Instrument's Datasheets and walks through the operation. When focusing on the output Y, the difference between the circuits is clear. The 7405 chip's output BJT's collector terminal is left unconnected within

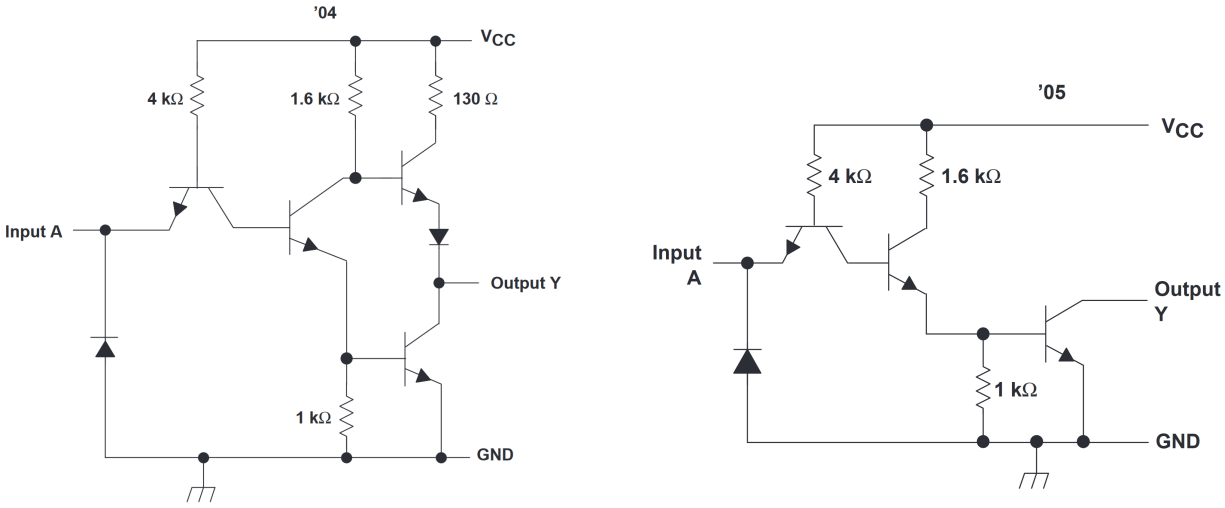
the chip. This is also known as an "open." This architecture allows the chip to be independent of the downstream current draw.

This makes it suitable for applications where a resistive load such as a motor, or in this case, an LED is connected to the output of the inverter. This does require some planning as the load requires a external pullup resistor in order to achieve a HIGH output.

There are three possible different configurations of a BJT circuit, a common emitter, common base, or a *common collector circuit* where "Common" references which two leads of the BJT are both part of the input and outputs circuits.

For the 7405, a positive-going input signal drives the base more positive with respect to the emitter, which causes the base current to increase. This results in more emitter current, causing a more positive output voltage at the emitter to drive our LED.

Example 2 ***Operation of the 7405***



(a) The Standard 7404

(b) The Standard 7405

Figure 2: Schematic View

When the input signal is high:

- *The base of the first NPN transistor is forward biased relative to its emitter, allowing base current to flow. This results in current flowing from the collector to the emitter, effectively grounding the output.*
- *The base current enables a larger current flow from the collector to the emitter, pulling the output low (close to ground).*
- *In this state, the transistor is "on," and the output is a logic LOW signal (approximately 0.0V to 0.8V in TTL logic).*

When the input signal is low:

- *The base of the first NPN transistor is not forward biased, so no current flows through the base-emitter junction. As a result, the collector-emitter path remains open, and the transistor is turned off.*
- *Since the transistor is off, the output floats and is pulled high by the external pull-up resistor, resulting in a logic HIGH output close to V_{CC}.*

In the 7405 open-collector configuration, the NPN transistor plays a crucial role in determining the output state based on the input signal.

As with any NPN transistor, the emitter, base, and collector function to control the current flow between the collector and emitter through the base current.

In this circuit, the NPN transistor's collector is left unconnected internally (hence the term "open collector"). The external load, such as an LED or a resistive element, is connected between the output (collector) and a pull-up resistor, which is tied to the supply voltage

In essence, the NPN transistor in the 7405 circuit acts as a switch that controls the flow of current to the output based on the base current. When there is base current, the transistor conducts, and the output is pulled low. When there is no base current, the transistor is off, and the output is pulled high by the external resistor.

This open-collector design is particularly useful for applications that require the driving of higher-current loads, such as LEDs, without requiring the IC itself to supply the current. Instead, the external load provides the necessary current, allowing the circuit to handle higher loads more efficiently.

To summarize, in plain non-electrical engineering speak, if the transistor is ON, current flows through the path of least resistance and robs the output (our LED) of its current, making the light go dim. If the transistor is OFF, the pullup resistor now makes the path of least resistance between the LED and ground and the LED illuminates.

Current Draw of Each LED

The current through each LED is calculated using Ohm's Law:

Example 3

$$I = \frac{V_R}{R} = \frac{V_{supply} - V_{LED}}{R}$$

Where:

- $V_{supply} = 5V$ *is the supply voltage.*
- V_{LED} *is the forward voltage of the LED. This is measured at* $1.786V$ *by my multimeter.*
- $R = 330\,\Omega$ *is the resistor value.*

$$I_{red} = \frac{5V - 1.786V}{330\,\Omega} = \frac{3.214V}{330\,\Omega} = 9.74\,mA$$

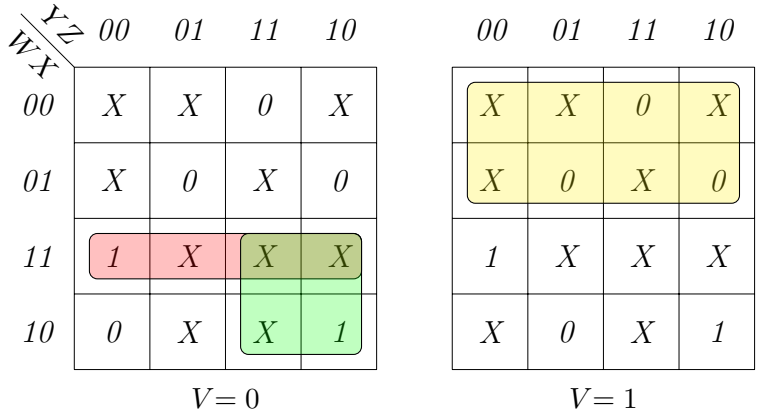
POSTNET to XS_3 Encoding

Decimal	V	W	X	Y	Z	D	C	B	A
0	1	1	0	0	0	0	0	1	1
1	0	0	0	1	1	0	1	0	0
2	0	0	1	0	1	0	1	0	1
3	0	0	1	1	0	0	1	1	0
4	0	1	0	0	1	0	1	1	1
5	0	1	0	1	0	1	0	0	0
6	0	1	1	0	0	1	0	0	1
7	1	0	0	0	1	1	0	1	0
8	1	0	0	1	0	1	0	1	1
9	1	0	1	0	0	1	1	0	0

Table 1: POSTNET to XS3 Conversion Table

Example 4 ***D Output***

The map is divided into two layers for $V = 0$ and $V = 1$. Pay Attention to V.



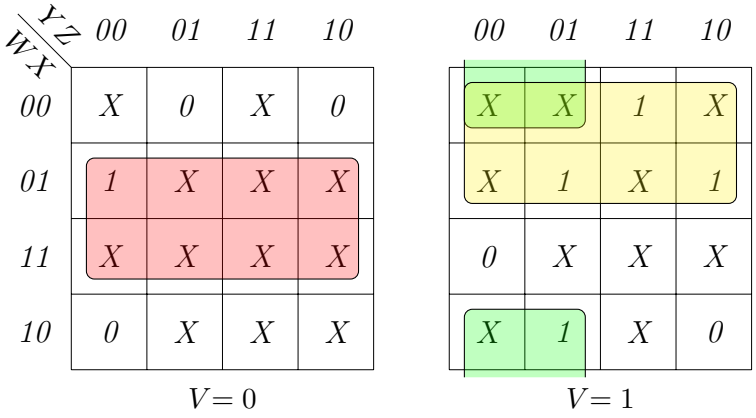
This produces:

$$D = V(\overline{W}) + \overline{V}(WX + WY)$$

$$D = V\overline{W} + \overline{V}WX + \overline{V}WY \quad \textit{(Distribution)}$$

$$D = V\overline{W} + WX + WY \quad \textit{(Absorption)}$$

Example 5 ***C Output***

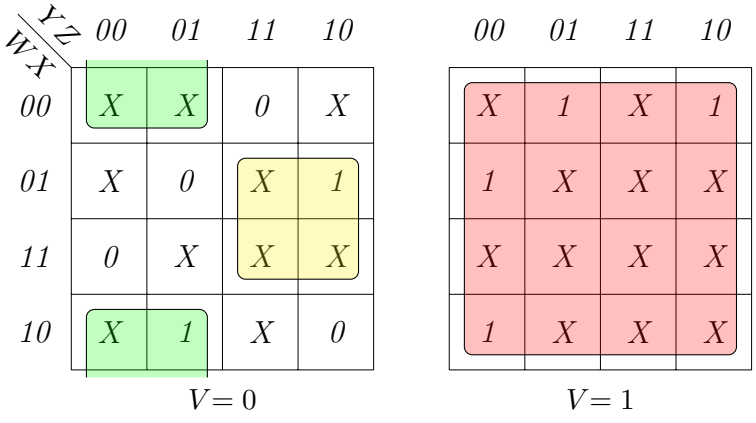


This produces:

$$C = \overline{V}(\overline{W} + \overline{X}Y) + V(X)$$

$$C = \overline{V}\overline{W} + \overline{V}\overline{X}Y + VX \quad \textit{(Distribution)}$$

Example 6 ***B Output***

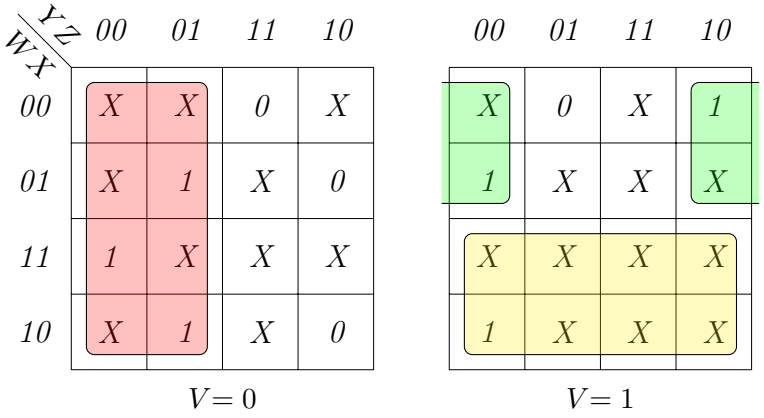


This produces:

$$B = V + \overline{V}(W\overline{Y} + XY)$$

$$B = V + \overline{V}W\overline{Y} + \overline{V}XY \quad \textit{(Distribution)}$$

Example 7 ***A Output***



This produces:

$$A = \overline{V}\overline{Y} + V(\overline{W}\overline{Z} + \overline{W})$$

$$A = \overline{V}\overline{Y} + V\overline{W}\overline{Z} + V\overline{W} \quad \textit{(Distribution)}$$

$$E = r \cdot \log_r N$$

where E is the economy, r is the radix (base) of the number system, and N is the number of unique values to be represented.

This metric balances two competing factors:

- The number of unique digits required (complexity)
- The number of digit positions needed to represent a given range of values (length)

A lower value of E indicates a more economical system, offering a better trade-off between these factors.

Implications and Applications

Systems with good radix economy can offer several advantages:

- Enhanced human readability
- Reduced system complexity
- More compact and efficient computational implementations
- Potential for reduced physical space requirements in hardware
- Improved circuit performance and reduced power consumption
- Potential for lower overall project costs in hardware design

Optimal Radix vs. Practical Considerations

Mathematically, the optimal radix for maximal economy is $e \approx 2.718$ which is derived by using the second derivative test to determine the maximum value. However, practical considerations often outweigh pure mathematical optimality

Binary (*radix-2*) systems are prevalent in computing despite not having the best radix economy. The simplicity of binary representation, with only two states (0 and 1), greatly simplifies computer architecture and logic design. This simplicity often outweighs the theoretical advantages of more economical bases in real-world applications.

The choice of number system thus involves a balance between theoretical efficiency and practical implementation constraints.

Deliverables

After calculating the minterms for our Z_0, Z_1, Z_2, Z_3 outputs, that is, the equations such that our corresponding output bit is TRUE (1), we get some fairly large equations that were then able to get reduced by boolean logic algebra. Unfortunately, because the lab limited us to only two-input OR, AND, and one-input NOT gates, the simplicity of the circuit was not able to get greatly reduced.

Example 8 *Outputs Z_3, Z_2, Z_1, Z_0:*

- $Z_0 = \overline{D}\overline{C}\overline{B}A + \overline{D}C\overline{B}\overline{A} + \overline{D}CBA + D\overline{C}B\overline{A}$
  $Z_0 = \overline{D}\overline{C}BA + CBA + CBA) + D\overline{C}B\overline{A}$
  $Z_0 = \overline{D}(\overline{C}\overline{B}A + C(\overline{B}\overline{A} \ + BA)) + D\overline{C}B\overline{A}$
  The Z_0 bit could be simplified by grouping the $\overline{D}$ signal to the outside of common multiples and some mild reduction utilizing distributivity and identity boolean algebra rules.
- $Z_1 = \overline{D}\overline{C}B\overline{A} + \overline{D}C\overline{B}A + D\overline{C}\overline{B}\overline{A}$
  $Z_1 = \overline{D}(\overline{C}B\overline{A} + C\overline{B}A) + D\overline{C}\overline{B}\overline{A}$
  The Z_1 bit could also be simplified by grouping the $\overline{D}$ signal to the outside of common multiples.
- $Z_2 = 0$ *(Always FALSE)*
- $Z_3 = 0$ *(Always FALSE)*

Discussion

The practical portion of the lab was more involved with Logicworks as we learned about simplifying Boolean expressions. The inclusion of the Hex Keyboard into the schematic allowed for a nicer way of interacting with the circuits and careful wire routing throughout the workspace was crucial for success in completing this. Ultimately, the simplifications couldn't be applied as much as I initially thought that would be possible when it came to designing the circuit, but nevertheless each network of signals was easily troubleshot and ran through until the target output was reached.

Practically, this was quite engaging as the network of wires needed to be very closely monitored to ensure that a certain system wouldn't change after modification. After some troubleshooting and careful analysis of wire and inverter placement, the circuit was able to run, and the desired truth table was output successfully.

To conclude and summarize, the Lab reinforced concepts learned in class through pen and paper arithmetic and familiarity with the Digital Design Software tools in Logicworks.

Practice Questions

Radix Conversion

Given the decimal number 92_{10} we can change between number systems utilizing the algorithm discussed previously.

Example 9 ***Binary:***

$$92_{10} = (1 \cdot 2^6) + (0 \cdot 2^5) + (1 \cdot 2^4) + (1 \cdot 2^3) + (1 \cdot 2^2) + (0 \cdot 2^1) + (0 \cdot 2^0)$$

$$92_{10} = 1011100_2$$

Example 10 ***Octal:***

$$92_{10} = (1 \cdot 8^2) + (3 \cdot 8^1) + (4 \cdot 8^0)$$

$$92_{10} = 134_8$$

Example 11 ***Hexadecimal:***

$$92_{10} = (5 \cdot 16^1) + (12 \cdot 16^0) \quad \textit{(where } C = 12\textit{)}$$

$$92_{10} = 5C_{16}$$

2's Compliment and Decimal Conversion

Given the signed binary number 11001101_2 swapping between 2's compliment and decimal is extremely easily. Begin by looking at the Most-Significant-Bit (MSB) for the first information about what decimal number this signed binary number contains. An MSB of 1 indicates a negative number, an MSB of 0 represents a positive number.

The two's complement representation facilitates simpler arithmetic operations and algorithms in digital systems. However, this comes at the cost of a reduced range for the maximum positive number that can be represented, compared to unsigned integers of the same bit length.

Example 12 ***Size limits***
Unsigned 8-bit integer:

- *Range:* $0 \rightarrow 2^8 - 1$ *(0 to 255)*

Signed 8-bit integer (two's complement):

- *Positive range (MSB = 0):* $0 \rightarrow 2^7 - 1$ *(0 to 127)*
- *Negative range (MSB = 1):* $-2^7 \rightarrow -1$ *(-128 to -1)*

Note: *The signed range includes one more negative number (-128) than positive number (127) due to the nature of two's complement representation.*

Example 13 ***Two's Complement Conversion:***
Flip the bits and add 1.

$$\begin{array}{rl} 11001101_2 & \textit{(original number)} \\ 00110010_2 & \textit{(flipped bits)} \\ +00000001_2 & \textit{(add 1)} \\ \hline 00110011_2 & \textit{(two's complement result)} \end{array}$$

The two's complement of 11001101_2 *is:*

$$00110011_2$$

Example 14 *Two's Complement to Decimal:*
To convert 11001101_2 *(signed) to decimal, we take the two's complement result and compute its negative value.*

The two's complement result is 00110011_2*, which is positive. Converting this binary number to decimal:*

$$00110011_2 = (0 \cdot 2^7) + (0 \cdot 2^6) + (1 \cdot 2^5) + (1 \cdot 2^4) + (0 \cdot 2^3) + (0 \cdot 2^2) + (1 \cdot 2^1) + (1 \cdot 2^0)$$

$$= (1 \cdot 32) + (1 \cdot 16) + (1 \cdot 2) + (1 \cdot 1) = 32 + 16 + 2 + 1 = 51$$

Since 11001101_2 *is a negative signed number, the decimal value of the original binary number is:*

$$-51_{10}$$

Boolean Logic to Gates

To finish things off, the extraction of 2 input gates from a simple boolean expression is done in the following example accompanied by the corresponding truth table.

In order to correctly populate the table, each expression was treated seperately and ORed together to form the final output column of F.

Example 15 $\overline{A} + C\overline{D} + BA$ *is expressed in logic gates in the following figure*

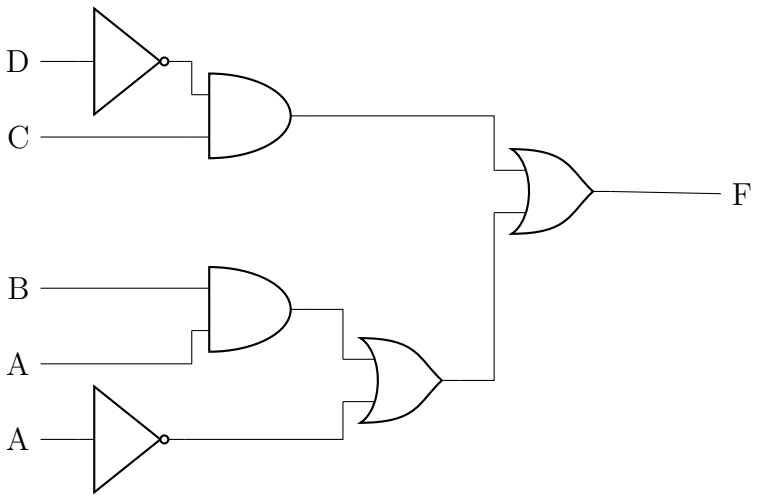


Figure 3: Logic gate diagram for $\overline{A} + C\overline{D} + BA$

D	C	B	A	$\overline{A}$	$C\overline{D}$	BA	F
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1
0	0	1	1	0	0	1	1
0	1	0	0	1	1	0	1
0	1	0	1	0	1	0	1
0	1	1	0	1	1	0	1
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	0	0	1	1
1	1	0	0	1	0	0	1
1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	1
1	1	1	1	0	0	1	1