Chapter 4:

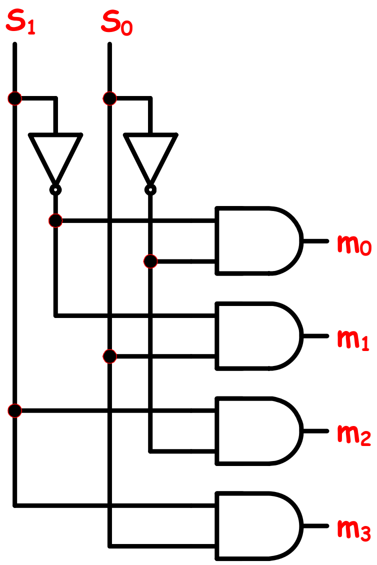
Table of Contents

Decoders

Table of Contents

Modular Combinational Logic - Decoders

The Generic Decoder



Generic 2/4 Decoder

A decoder is a **min-term generator** with **each output corresponding to a single min-term**. They are generally used for code conversions (**binary** to **decimal**), data routing, or equation creation. They are also referred to as "**line decoders**" due to the fact that **the user can "activate" an output line by specifying a "control word".**

The ***generic*** **discrete 2/4 decoder** in the figure to the right has **active high inputs** and **outputs**. Each output (or min-term) in the circuit is produced by **AND'ing** the specified control signals S_1, $\overline{S_1}$, S_0, ***or*** $\overline{S_0}$, which results in a unique output. It should be noted that there is **absolutely NO way that more than one output can be active at the same time.** The terminology **2/4** indicates that **two inputs decode into four outputs**.

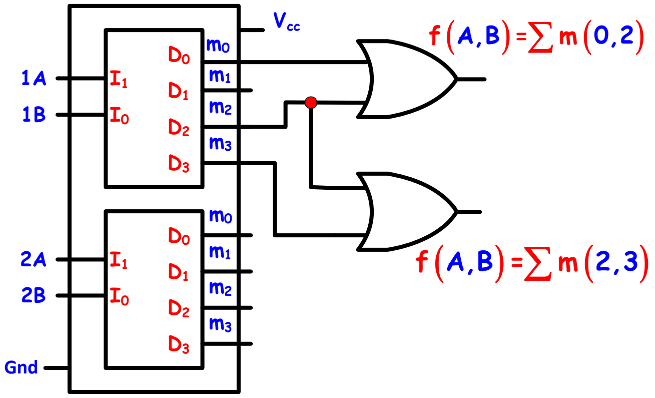
As stated earlier, a **decoder** will **ALWAYS have ONE and ONLY ONE ACTIVE output at a time.** Normally, you will have two **2:4 Decoders** on a single 14 pin chip. As is demonstrated in the circuit below, by combining the outputs with '**OR**' gates you can create unique **SOP** equations.

It should be obvious to the student that the fewer the number of chips it takes to produce a logic expression:

- the lower **real-estate** on the printed circuit board,
- the lower **power dissipation,**
- the lower the **manufacturing costs**,
- fewer chips means **higher reliability**.

Generic Example:

In the example circuit to the right, a **generic 2-4 decoder chip** has one of its decoders being used to create two separate switching expressions.

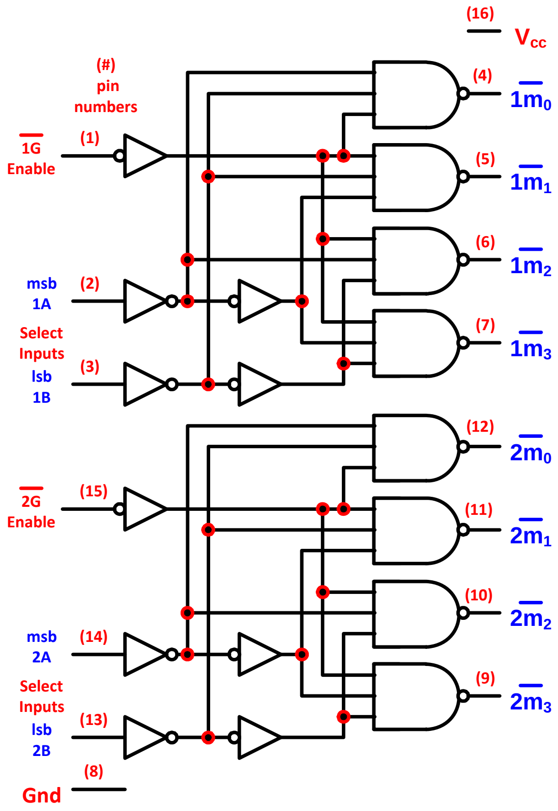


Thus far, we have only discussed generic decoders with **active high** inputs/outputs. There are several popular decoders on the market which utilize **active high** and/or **active low inputs and outputs**. One such device is the **74139** as discussed next.

The 74139 decoder

The **74139** is an excellent example of a **dual 2/4 decoder chip**. It features:

- ***Active High Inputs***
- ***Active Low Outputs***
- ***Active Low Enable Input***



As can be seen in the circuit to the right, the **enable, (G),** goes to one input on each of the NAND gates. **There is no way that anything except a high can be seen at the outputs unless this enable input is Active, (0).** Therefore, if the device is **disabled, all outputs go high** since they are **active low** outputs. If they were **active high** outputs, they would go low **when disabled**.

Note the **additional inverter** in the select inputs shown in the diagram:

Question: **What does it do?**

Answer: **It improves the fan-in.** To the outside world, it looks like one gate vice three gates, which is what would be seen without the inverter.

The Decoder Specification sheet

Now let's examine a few entries on the **74139** specification sheet to the right.

	SN74LS139A			
	MIN	NOM	MAX	Unit
Supply Voltage	4.75	5	5.25	V
High-level input voltage	2			V
Low-level input voltage			0.8	V
High-level output current			-0.4	mA
Low-level output current			8	mA

- Note the difference between the **maximum** supply **voltage** and the **recommended** supply **voltage**.
  - This tends to be fairly standard with **TTL** devices.

- The table also indicates that a
  - **"HIGH"** is **2v or greater** **(High-Level Input Voltage)**
  - **"LOW"** is **< .8v** **(Low-Level Input Voltage)**

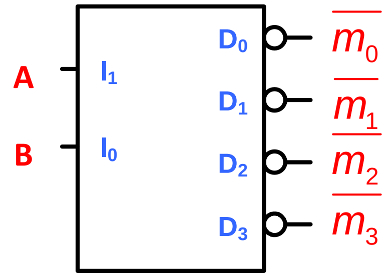
- The **high** and **low output currents** are also given. However, as long as you **don't exceed the fan-out**, you don't have to worry about these items. There is one other item which is very important.
  - The **Short Circuit Output Current - No more than one output should be shorted or you could exceed this number.**
    - Again, this should not come as a surprise that if you **SHORT outputs together that they will at the very least be degraded.**

2/4 Decoder Examples

Example 4-4a

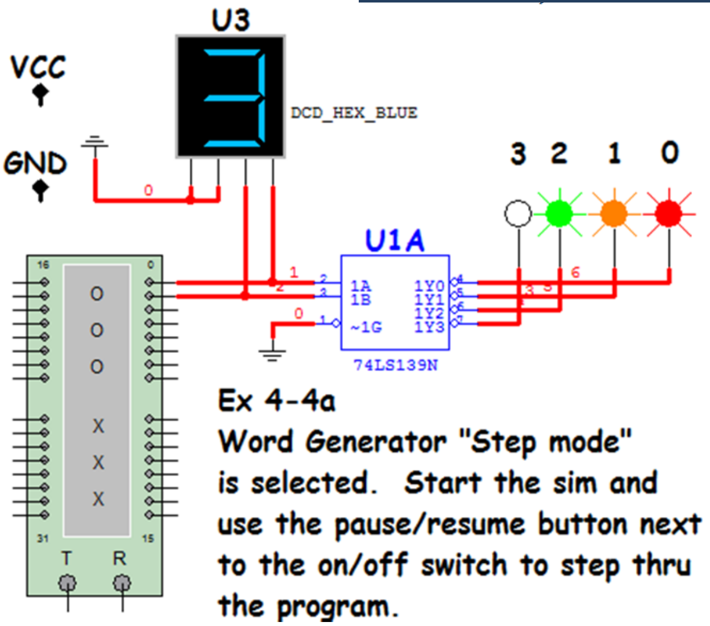
Let’s take a look at a decoder with **active low outputs**:

Example 4-4a

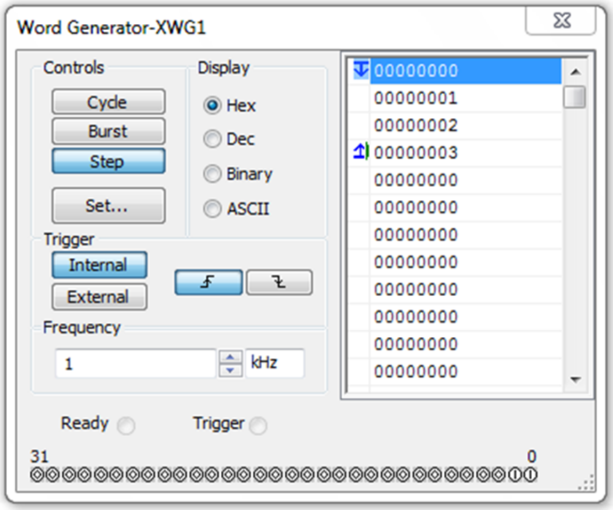


I_1	I_0	D_3	D_2	D_1	D_0
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

Multisim Implementation of Example 4-4a



Ex 4-4a
Word Generator "Step mode" is selected. Start the sim and use the pause/resume button next to the on/off switch to step thru the program.


Word Generator-XWG1

Controls: Cycle, Burst, Step, Set...

Display: ● Hex ○ Dec ○ Binary ○ ASCII

Trigger: Internal, External

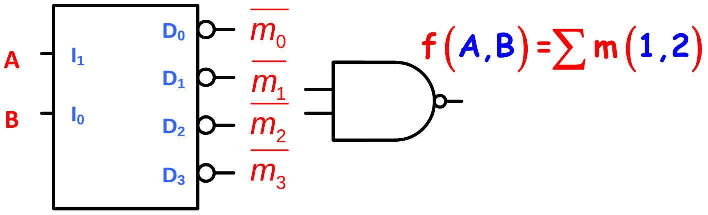
Frequency: 1 kHz

```
00000000
00000001
00000002
00000003
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
```

Ready ○ Trigger ○

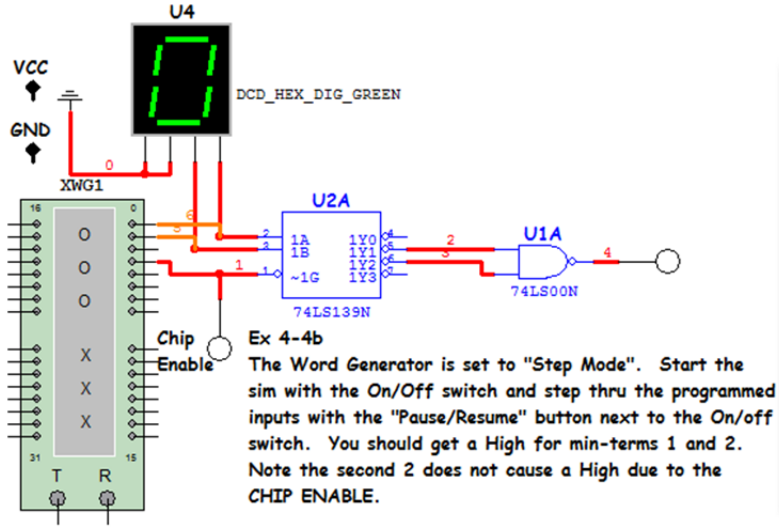
Example 4-4b

$$f(A,B) = \overline{\overline{m_1} * \overline{m_2}}\ (NAND)$$
$$= \overline{\overline{m_1}} + \overline{\overline{m_2}} = m1 + m2$$
$$= \sum m(1,2)$$

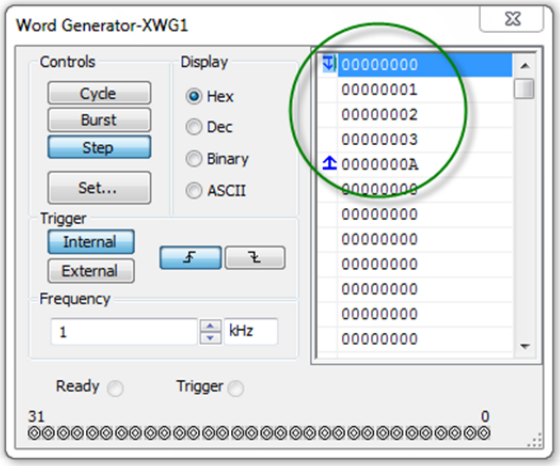


MultiSim implementation of 4-4b

The Word Generator is programmed as before. Note that this time the **Chip Enable** has been tied to **Bit - 2 of the Word Generator which is always 0 in this example. Thus, it acts the same as tying the input to ground.**



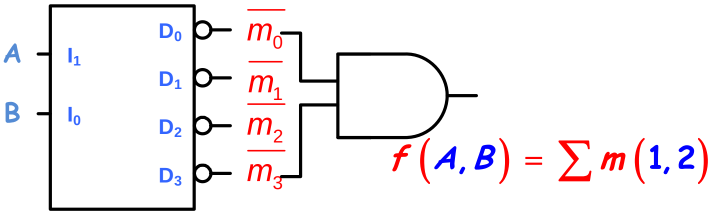
Ex 4-4b
The Word Generator is set to "Step Mode". Start the sim with the On/Off switch and step thru the programmed inputs with the "Pause/Resume" button next to the On/off switch. You should get a High for min-terms 1 and 2. Note the second 2 does not cause a High due to the CHIP ENABLE.



```
00000000
00000001
00000002
00000003
0000000A
00000000
00000000
00000000
00000000
00000000
00000000
00000000
```

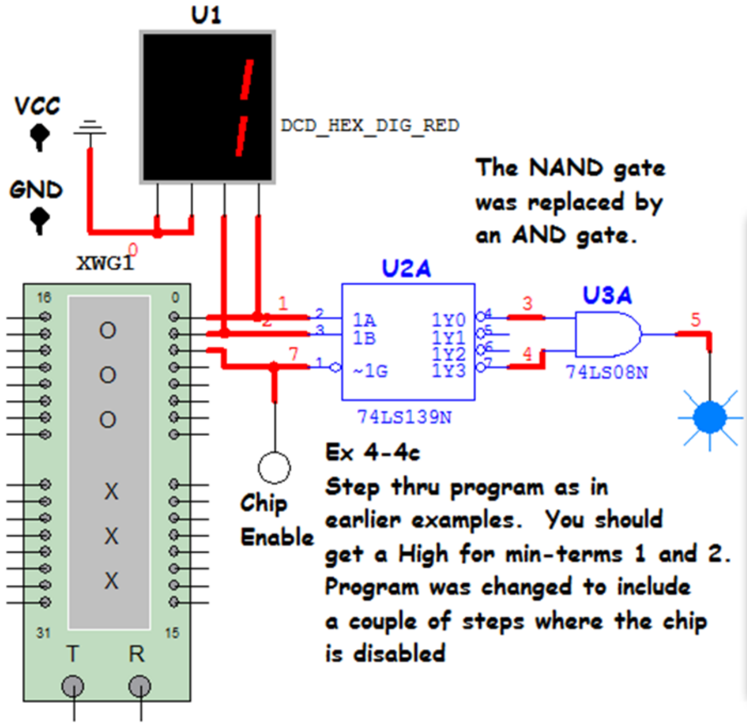
Example 4-4c

Note that we can achieve the same results as the last example if we use a **AND** gate vice a **NAND** gate but we need to attach to outputs **0** and **3** instead.



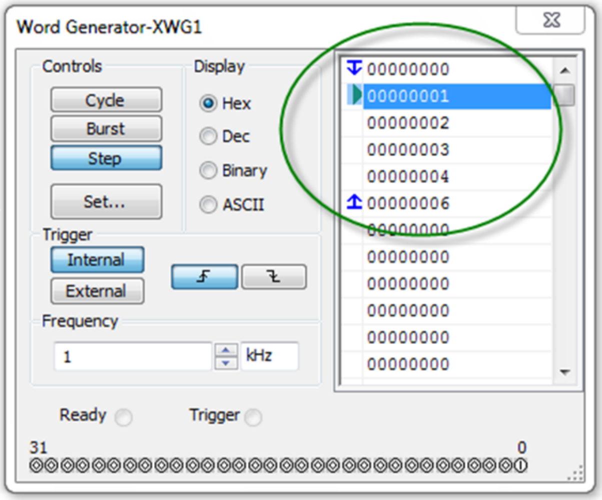
$$f(A,B) = \sum m(1,2)$$

MultiSim implementation of 4-4c



The NAND gate was replaced by an AND gate.

Ex 4-4c
Step thru program as in earlier examples. You should get a High for min-terms 1 and 2. Program was changed to include a couple of steps where the chip is disabled



Display: ● Hex ○ Dec ○ Binary ○ ASCII

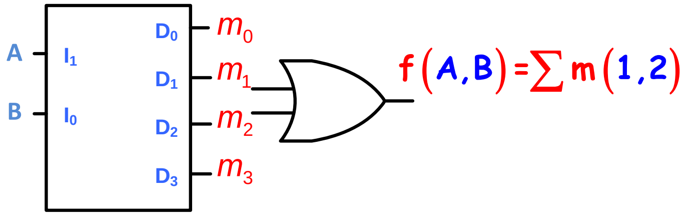
Frequency: 1 kHz

```
00000000
00000001
00000002
00000003
00000004
00000006
00000000
00000000
00000000
00000000
00000000
00000000
```

A decoder with active High outputs

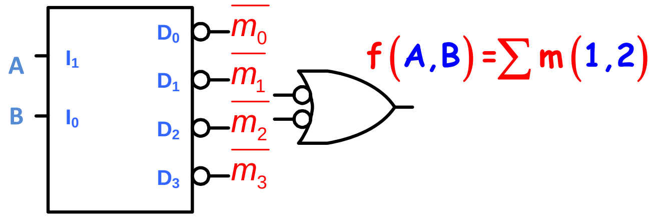
Now let's look at a decoder with **Active High Outputs** compared with one with **Active Low Outputs.** The **circuit above** demonstrates that we can **AND** the **ACTIVE LOW outputs that are not in our min-term list** and get the **desired min-term list**. Note the difference in this circuit and the one on the previous page.

The circuit below has **ACTIVE HIGH** outputs. Note that we can use the more obvious **OR gate** for the solution.



A Negative Logic Solution

ACTIVE LOW outputs sometimes give people trouble. Let's look at the same thing now but use **negative logic** to help clear up the situation.

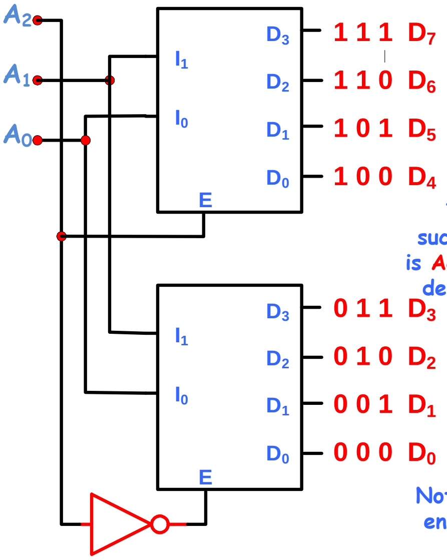


Remember that an **SOP expression** is a **SUMMATION of individual min-terms**. Also remember that when we combine positive and negative logic, we can **cancel out MATCHED Bubbles**. So, if the matched bubbles are canceled we can see that we really do have: $f(A,B) = \sum m(1,2)$.

Unfortunately, **<u>MultiSim doesn't have Negative logic gates</u>**. This is just a method of analyzing the circuit so that it means something.

The 3/8 decoder

Now, let's demonstrate how we can use **two 2/4 decoders to build a single 3/8 decoder.** This conversion is performed with the addition of an inverter to the circuit. As can be seen below, when one device is active, the other will be inactive. If we make the input to the Enable's the **MSB** of the **input control word**, we now have **<u>three inputs decoded to eight outputs</u>**.

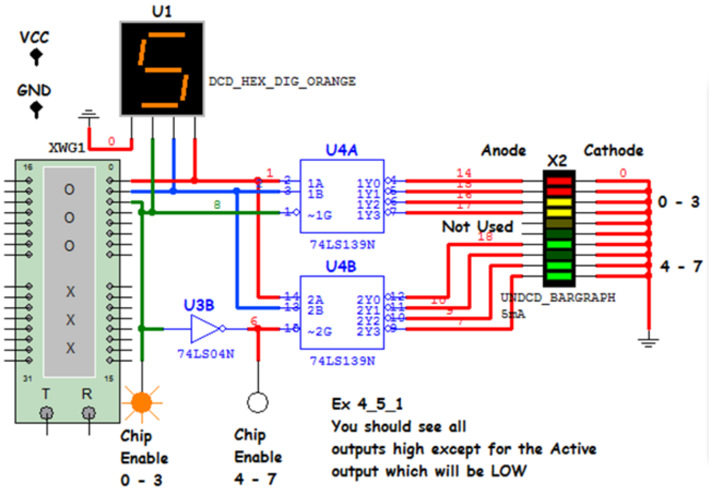


This decoder is designed such that the bottom decoder is Active for 0 to 3 and the top decoder is active for 4 to 7.

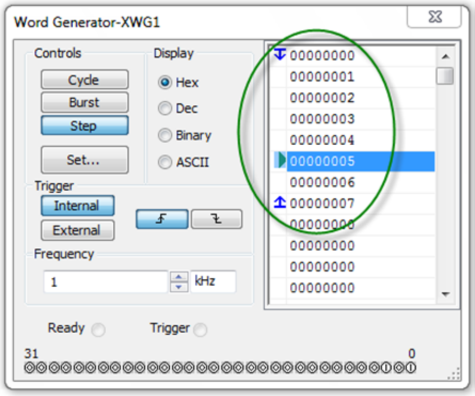
Two 2:4 decoders equate to One 3:8 decoder.

Note: For this example, the enables are ACTIVE HIGH.

<u>MultiSim Example of building a 3/8 decoder</u>



Ex 4_5_1
You should see all outputs high except for the Active output which will be LOW



Display: ● Hex ○ Dec ○ Binary ○ ASCII

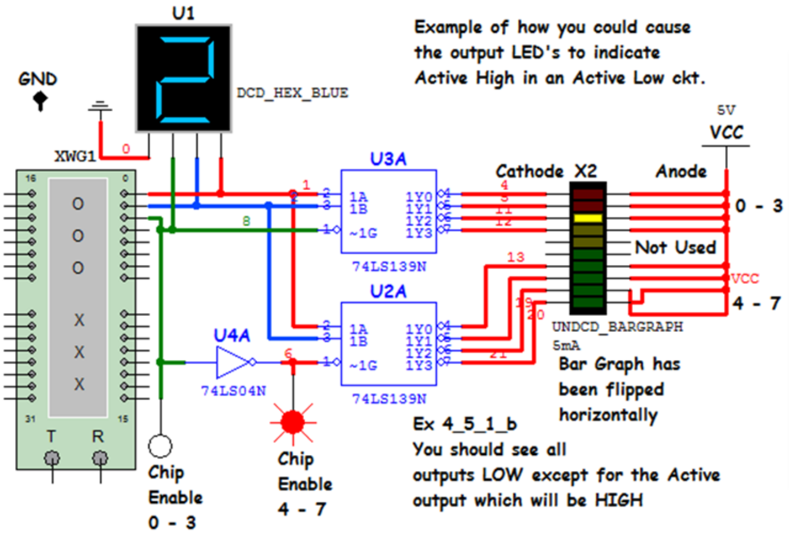
Frequency: 1 kHz

00000000
00000001
00000002
00000003
00000004
00000005
00000006
00000007
00000000
00000000
00000000
00000000

Note that so far we have had active low output displays for active low outputs. The following is an example of how we can cause the display to display active high when we have active low outputs.

When the Bar graph was placed in the circuit, it was **flipped horizontally so that the cathode and anode were switched**. Then the anode was attached to V_{cc} vice ground.

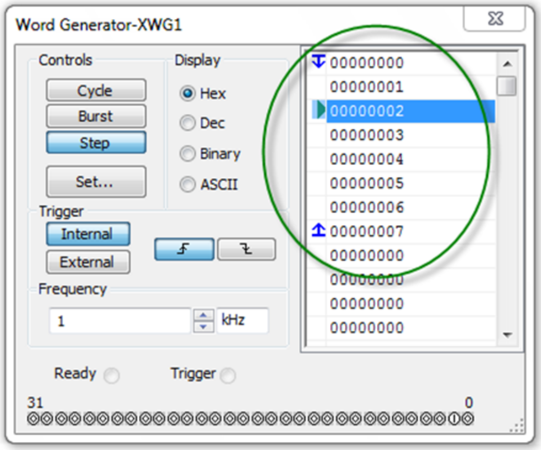
Multisim Example: Making Active Low look like Active High



Example of how you could cause the output LED's to indicate Active High in an Active Low ckt.

Bar Graph has been flipped horizontally

Ex 4_5_1_b
You should see all outputs LOW except for the Active output which will be HIGH


Word Generator-XWG1

Controls: Cycle | Burst | Step | Set...

Display: ● Hex ○ Dec ○ Binary ○ ASCII

Trigger: Internal | External

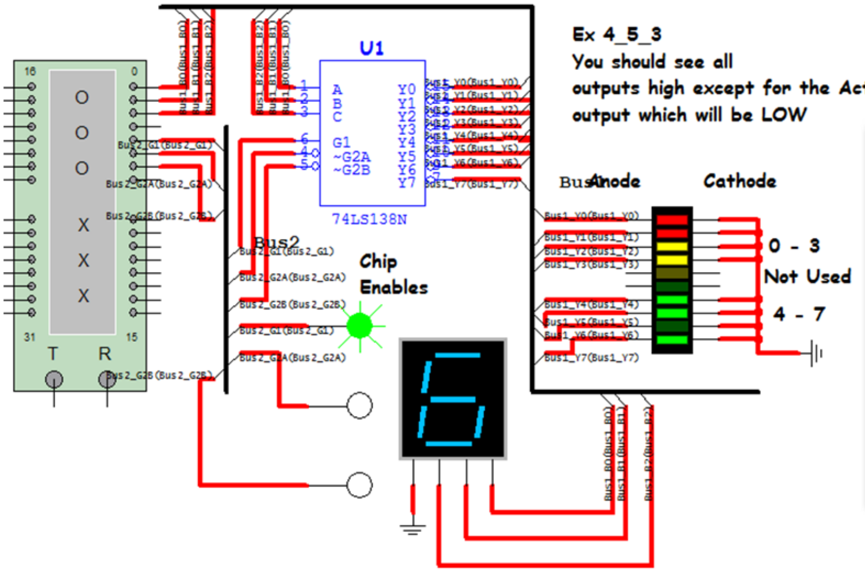
Frequency: 1 kHz

Ready ○ Trigger ○

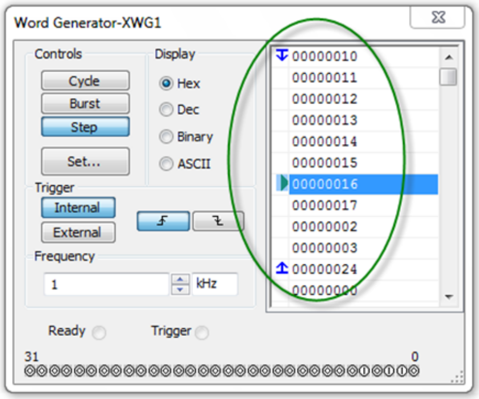
```
00000000
00000001
00000002
00000003
00000004
00000005
00000006
00000007
00000000
00000000
00000000
00000000
```

A MultiSim realization of a 74138 (using Busses)

In this example the chip is **enabled until the last 3 program steps**.



Ex 4_5_3
You should see all outputs high except for the Active output which will be LOW


Word Generator-XWG1

Controls: Cycle | Burst | Step | Set...

Display: ● Hex ○ Dec ○ Binary ○ ASCII

Trigger: Internal | External

Frequency: 1 kHz

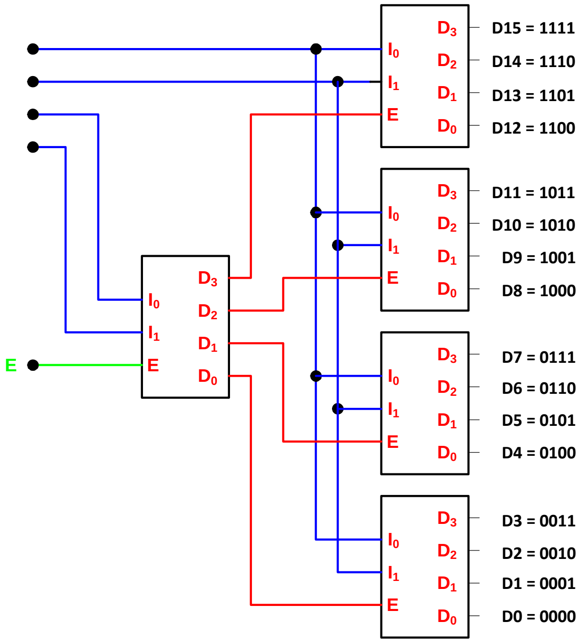
Ready ○ Trigger ○

```
00000010
00000011
00000012
00000013
00000014
00000015
00000016
00000017
00000002
00000003
00000024
00000000
```

The 4/16 decoder

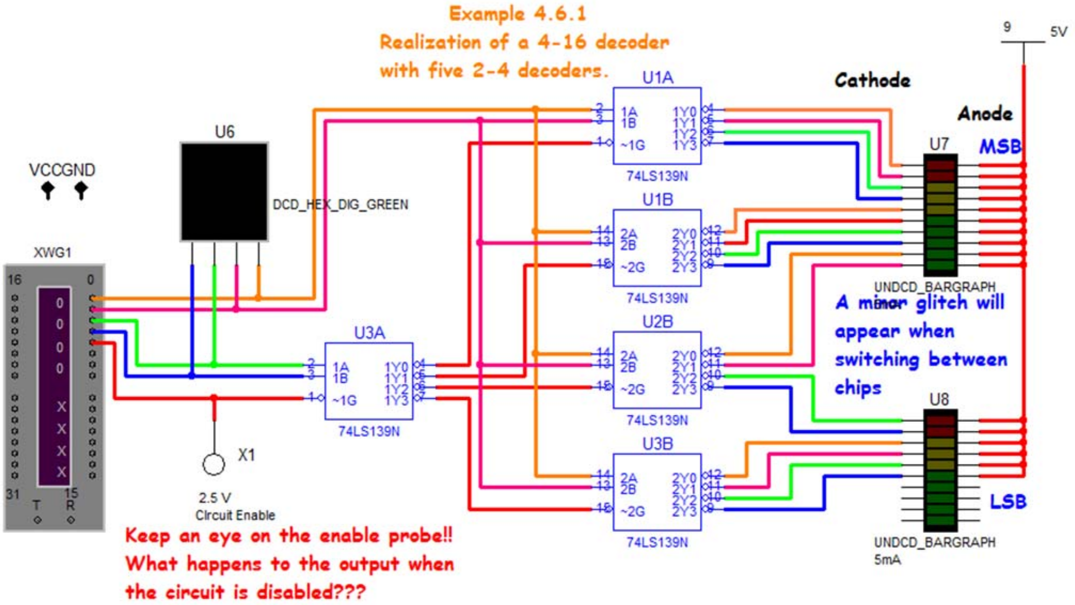
Let’s take a look at an even larger decoder. We can create a **4/16 decoder** using **five 2/4 decoders.**

In the figure, a **fifth decoder** is used to **select** which of the **four other decoders is active.**



A MultiSim Realization of a 5 chip 4/16 decoder

In the example below, remember that the outputs have been inverted to appear like they are active high by reversing the LED’s at the output and tying them to V_{cc} vice ground.

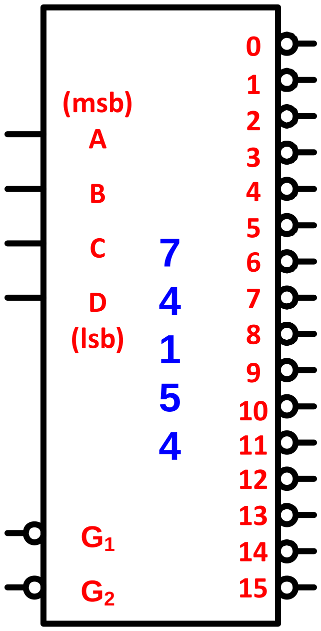

Example 4.6.1
Realization of a 4-16 decoder with five 2-4 decoders.

A minor glitch will appear when switching between chips

Keep an eye on the enable probe!! What happens to the output when the circuit is disabled???

The 74154 4/16 decoder

The **74154** is an example of a popular "off-the-shelf" 4/16 decoder. It features **active high inputs** and **active low outputs**, with **two active low enable inputs**.

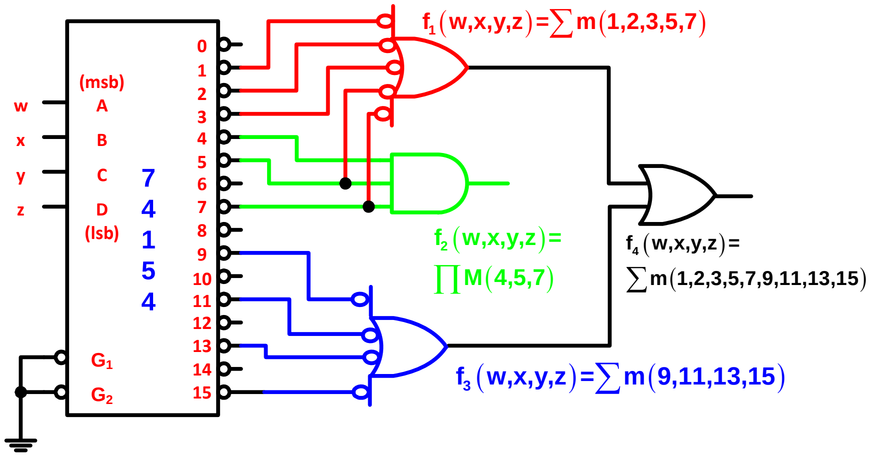


Question: **While the 74154 is a very popular decoder chip, what are the advantages to using the five 2/4 decoders option instead?**

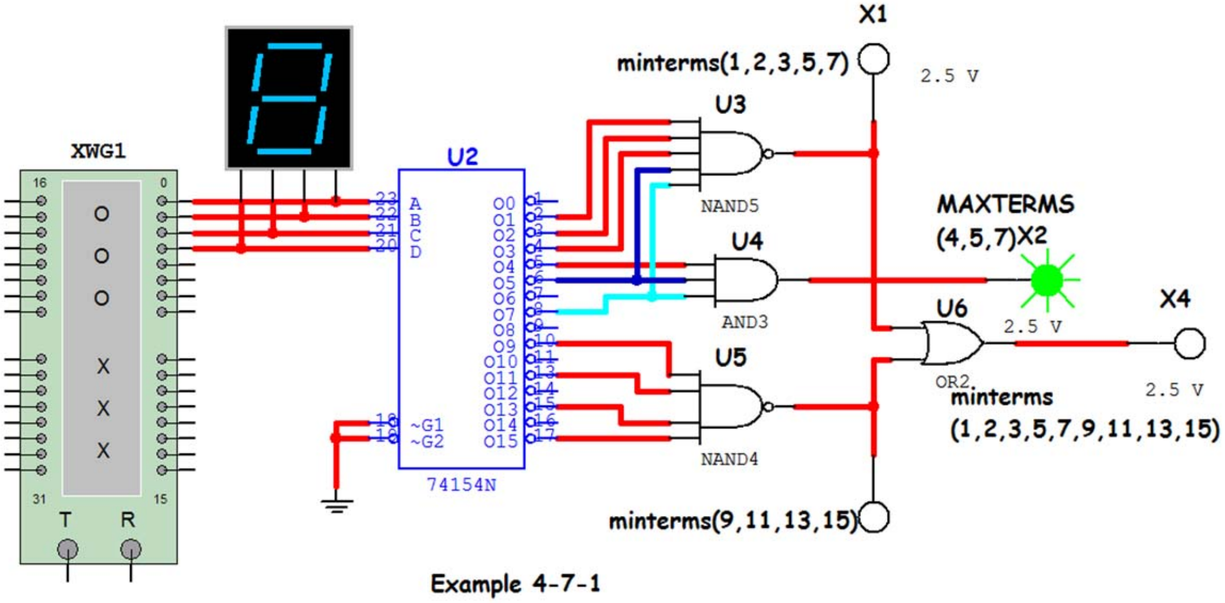
Answer: **The 4/16 Decoder chip is a 24 pin dip with a 0.6" center vice the 0.3" center for the 2/4 decoder 16 pin dip. If the 4/16 decoder chip was the only 24 pin chip on the PC board, the price of the completed board might be cheaper if the designer chose to use the 0.3" center devices instead.**

Let's look at the same example, but this time we used mixed logic to see if it makes the resulting expressions any clearer.

Remember that in mixed logic, if you can match bubbles, the bubbles cancel out.



MultiSim Example of a 4 to 16 decoder implementation



Example 4-7-1

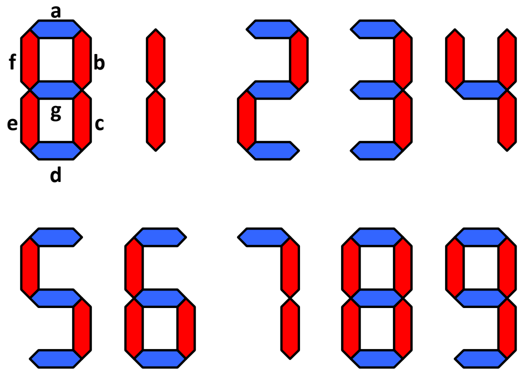
Decoder Case Study #1: BCD to Decimal Decoder

Of course, we have already designed a BCD to Decimal Decoder out of gates with the use of K-maps and 'Don't Cares' in a previous chapter. But we could use just a **74154 4:16 decoder** to do the same job. Since the BCD numbers are equivalent to decimal numbers from 0-9, all we have to do is use the **74154** and ignore the outputs 10-15. This may be more expensive than using the cheaper gates but it **might save money** in the long run due to:

- **cheaper construction cost and**
- **lower real estate taken up on the PC board.**

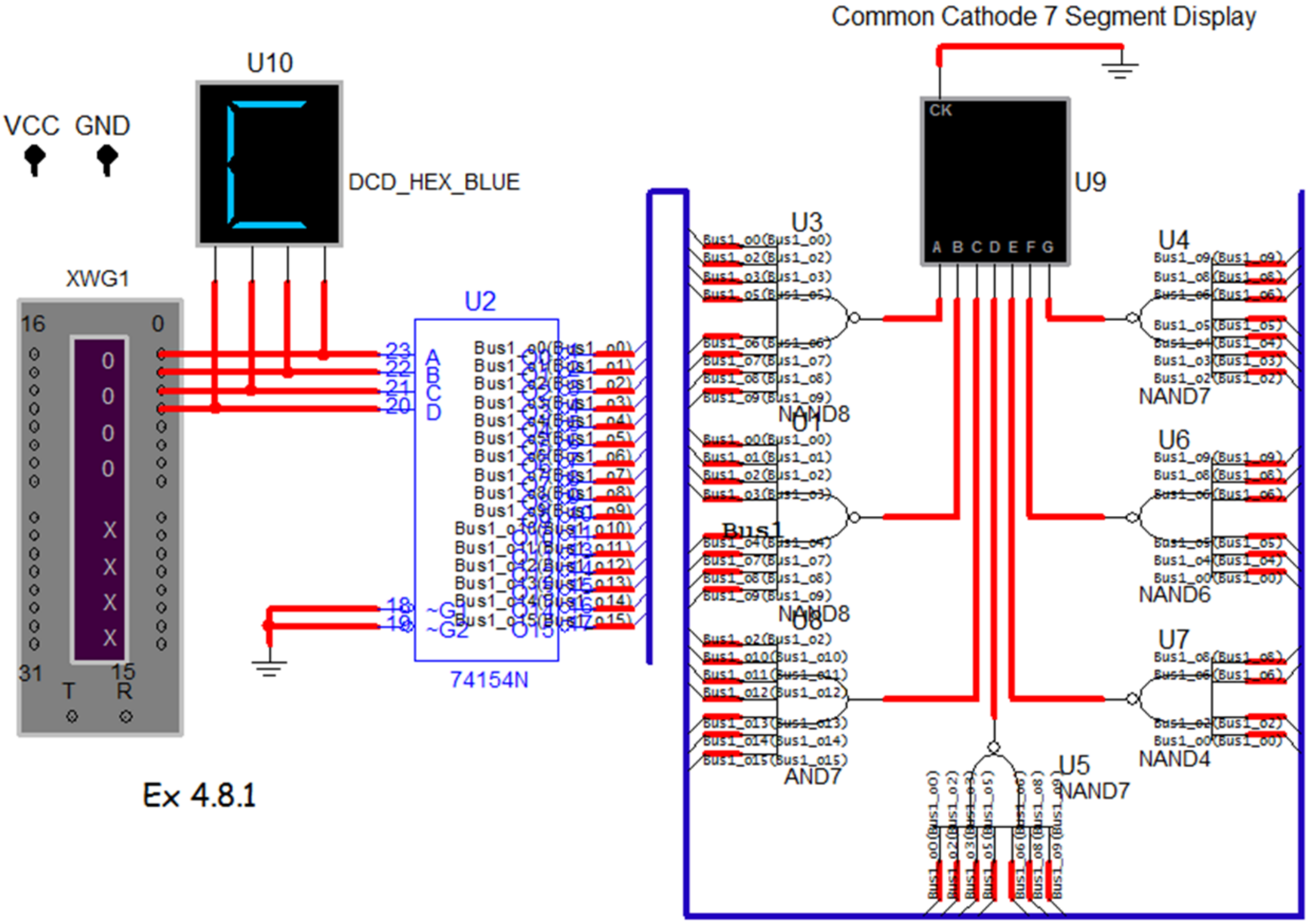
And don't forget that the output of the 74154 is **negative logic** so you would have to take that into account.

BCD input				Seven-Segment Decoder						
D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
All other inputs				0	0	0	0	0	0	0



Gate Type and Size if a 74154 is used	Switching List (Fan-in minimized)	Switching List (Fan-in minimized)	Gate Type and Size if a 74154 is used
NAND8	$F_a(D,C,B,A) = \sum m(0,2,3,5,6,7,8,9)$	$F_d(D,C,B,A) = \sum m(0,2,3,5,6,8,9)$	NAND7
NAND8	$F_b(D,C,B,A) = \sum m(0-4,7-9)$	$F_e(D,C,B,A) = \sum m(0,2,6,8)$	NAND4
AND7	$F_c(D,C,B,A) = \sum m(0,1,3-9)$ $= \prod M(2,10-15)$	$F_f(D,C,B,A) = \sum m(0,4-6,8,9)$	NAND6
		$F_g(D,C,B,A) = \sum m(2-6,8,9)$	NAND7

Multisim Implementation of a BCD to Decimal Decoder



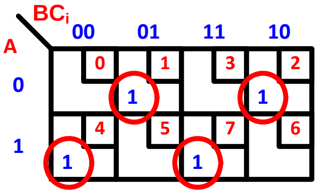
(Note that the 7-segment display is a "common cathode" or "CK" type. In order for it to work in Multisim it SOMETIMES has to have the CK input grounded thru a 75 ohm resistor.)

Decoder Case Study #2: Implementing a Binary Adder with a Decoder

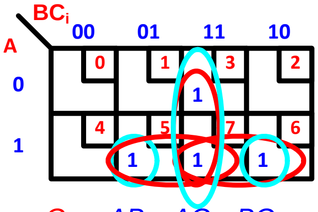
Contemporary Approach

The truth table for a full adder is as follows:

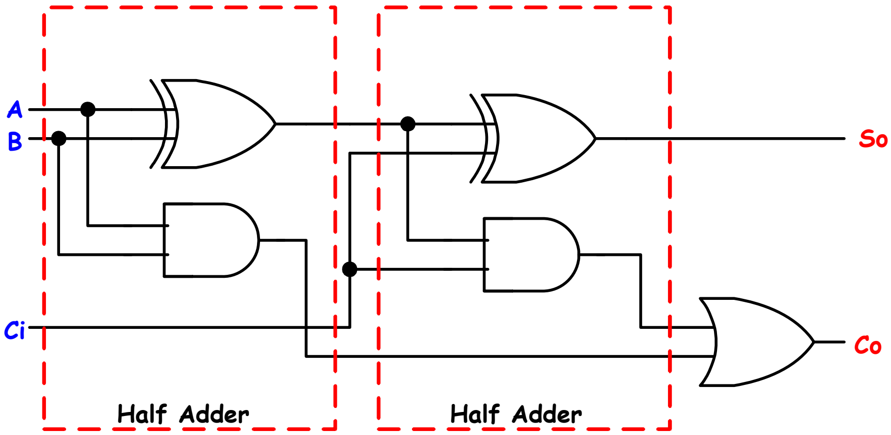
A	B	C_i	S_o	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$$S_o = \overline{A}\,\overline{B}C_i + \overline{A}B\overline{C_i} + A\overline{B}\,\overline{C_i} + ABC_i$$
$$= A \oplus B \oplus C_i$$



$$C_o = AB + AC + BC$$
$$= AB + C_i(A\overline{B} + \overline{A}B)$$
$$= AB + C_i(A \oplus B)$$

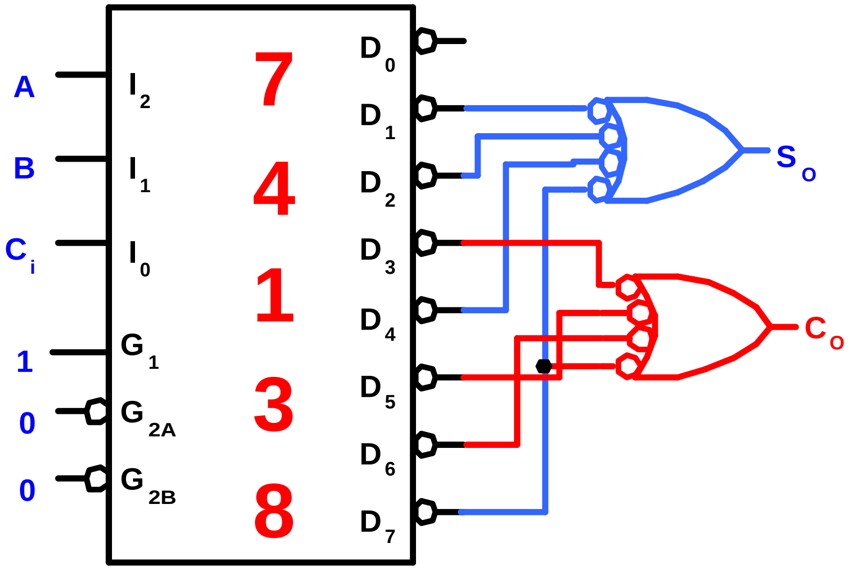


Decoder Approach

From the Full Adder table on the previous page, we can derive the following min-term lists:

$$S_o(A,B,C_i)=\sum m(1,2,4,7)$$
$$C_o(A,B,C_i)=\sum m(3,5,6,7)$$

Note that we used the negative logic NAND gates to view this representation. We used a total of 2 chips to implement this circuit while the contemporary method used 3 chips (1 XOR, 1 AND, 1 OR).


A
B
C i
1
0
0
I 2
I 1
I 0
G 1
G 2A
G 2B
74138
D 0
D 1
D 2
D 3
D 4
D 5
D 6
D 7
S o
C o