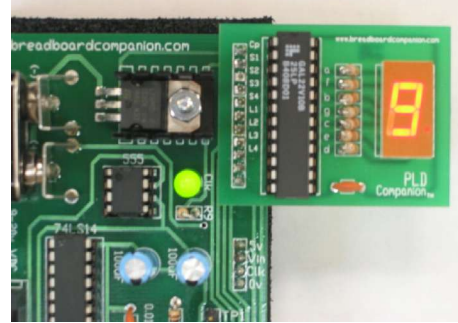




www.breadboardcompanion.com



BBCII Power Supply Kit and PLD Companion

Project Ideas

The PLD Companion™ prototyping board transforms the Breadboard Companion™ power supply kit (BBCII, or BBCJr) into a complete circuit-testing package. With the aid of a PLD, project ideas are endless. This document was written for a wide range of readers. For those who have little to no experience with PLDs, this document will give a breakdown on how to obtain, use and program a PLD. For those who are well acquainted with PLDs, this document will give suggestions of how to use the PLD Companion hardware with your existing PLDs.

Contents

1. What is a PLD?

- A. The Process of Designing a Digital Circuit (2)
- B. What you will Need (4)
- C. The Pins of a PLD (6)
- D. The Process of Programming a PLD (7)

2. Combinational Logic Projects

- A. Parking Space Problems (11)
- B. Binary to Hexadecimal Decoder (13)
- C. Sixteen Character Phrase (15)

3. Sequential Logic Projects

- A. 0000-1111 Binary Counter (17)
- B. Automated Phrase or Numeric Counter (19)

4. Final Project (21)

1. What is a PLD?

A. The Process of Designing a Digital Circuit

A PLD is a programmable chip for digital circuits. It stands for *Programmable Logic Device* and it allows an entire digital circuits to fit inside of one chip. In order for us to use a PLD, therefore, we have to know how to design digital circuits. In this section, we will describe the basic concept of a digital circuit and what is involved in designing one. For those who has little to no experience with designing a digital circuit, it would be recommended to consult a traditional textbook on digital electronics to augment what is discussed in this document. *The Science of Electronics: Digital* published by Pearson Prentice Hall has some excellent material on PLDs in addition to the traditional topics of digital logic.

In order to program a PLD, you must answer three very important questions.

1. What are my inputs?
2. What are my outputs?
3. How is one related to the other?

When we go to program a PLD, we will have to identify our inputs and outputs by giving them names and assigning pin numbers to them. This way, the PLD knows *where* to connect the pins together on the inside of the chip and we know how to connect the input and output hardware to the outside of the PLD. Secondly, we will have to identify how the inputs and outputs are related so that the PLD knows *how* to connect the pins together.

With this in mind, every digital circuit we will build will involve the identification of our inputs, our outputs, and the relationship between the two. Once we decide what circuit we wish to build, we will begin by assigning names (variables) to the *inputs* and the *outputs*. The next step will involve building a *Truth Table*. From the truth table, we will develop *Unsimplified Boolean Equations*, and from the unsimplified equations, we can (though it is not necessary) develop *Simplified Boolean Equations*.

Inputs—Every circuit must complete some function. The function of a digital circuit will always involve the manipulation of something that is put into it. Inputs are devices that we have direct control over. For the purposes of our PLD, our inputs will consist of one or more switches, a regulated clock pulse, or both. In any of the cases, each individual input can only be in one of two states (hence the word *digital*), “off” or “on”. These terms can be misleading, so we will usually refer to their states as '1' or '0' and it is up to the designer of the circuit to decide what a '1' or a '0' means for each individual input. For our purposes, '1' will mean that the switch is in the position closest to the numbers on the dip switch (5V). On

the clock pulse, however, though it is alternating between '1' and '0', we will not see much of a difference between the two. For us, the significance of the clock lies in the frequency in which it alternates between these states. This frequency (controlled by the potentiometer on the Breadboard Companion), will determine the rate at which our sequential logic circuits change states.

Outputs—Since every digital circuit manipulates an input, we must at some point interact with this manipulated input. This is what an output does. Outputs are not controlled directly, but determined by the circuit and the inputs. For our purposes, outputs will always be LEDs or *Light Emitting Diodes*. We can see the output as a visible light turning “on” or “off” (hence the name *digital*). Though it is obvious what “on” means for an LED, it does not necessarily mean '1'. The reason for this is that '1' is *always* associated with 5v and, though counterintuitive, 5v does not always turn an LED “on”. To turn an LED on, you must have a voltage *difference* of 5v across its two ends. If 5v is applied to both sides, then you have a difference of 0v and the LED does not turn on. Therefore, '1' for an output sometimes means “on” and sometimes means “off.” For our projects, we will be using either the four red LEDs on the power supply kit *or* the 7-segment display on PLD Companion. Below is a list of how each works.

Four LEDs

- turn “on” when 5v is applied to each
- often referred to as “active high”
- '1' means “on”
- '0' means “off”

7-segment Displayed

- turn “on” when 0v is applied to each
- often referred to as “active low”
- '0' means “on”
- '1' means “off”

Truth Table—A table that describes the relationship between the inputs and outputs. Inputs are listed on the left and outputs are listed on the right. Since it is a digital circuit, the possible states of the inputs are only '1' and '0' and therefore all the possible input states are limited. Each row of the truth table is a possible state of the inputs and we exhaust all of these possibilities by counting in binary. We then fill in the outputs based on what we want the circuit to do. Examine project 2A, “Parking Space Problem” on page 10 for an example of a four input truth table with four outputs.

Unsimplified Boolean Equations—These are algebraic descriptions of what we see in the truth table. The equations are written *for* the outputs and *in terms of* the inputs. This means that you will have an equation for every output that you have.

Each output depends on the state of the inputs and that is why the output is written as the *dependent* variable and the inputs as the *independent* variables. Every time an output is '1', we need to write a product expression for that scenario. This product expression must include which inputs are '1' and which inputs are '0' for that particular row. We complete the equation by “adding” up all the product expressions. Below is an example of a simple circuit.

Inputs		Output		
<i>a</i>	<i>b</i>	<i>x</i>		
0	0	0		
0	1	1	→	$\neg a \& b$ meaning— “a is zero <i>and</i> b is one”
1	0	0		
1	1	1	→	$a \& b$ meaning— “a is one <i>and</i> b is one”

$x = \neg a \& b \# a \& b$	<i>Unsimplified Boolean Equation</i>
-----------------------------	--------------------------------------

meaning— “x will be one when a is zero *and* b is one
or when a is one *and* b is one”

Though the notation (i.e. '#' instead of '+') is not one that you would find in textbooks, it is the syntax for the software we will be using.

Simplified Boolean Equation—There are many ways to simplify a Boolean Equation, none of which are necessary if using a PLD. The software we will use to program our PLDs will automatically simplify the equations for us and, even if it did not, most of our unsimplified circuits would still fit inside of a single PLD. Most of the projects listed in this document give both the unsimplified and the simplified forms of the boolean equations. For those who wish to practice their simplification techniques or for those who just want to deal with simpler equations, the simplified forms are listed. For our example circuit, the simplified equation would be $x=b$; if you re-examine the truth table, you will see that every time b is one, x is one, and every time b is zero, x is zero. Consult a textbook on digital electronics for a breakdown on how the simplification process works.

B. What you will Need

In this section, you will find a list of everything you need in order to program PLDs and use them in designing digital circuits. All of the pin numbers and algebraic syntax given in the projects of this document will be in terms of the items listed in this section. Though most information should be accurate, the

listed prices are subject to change.

Keep in mind that there are many hardware and software options beyond those listed here. In order to keep it simple for the beginner, we will list only one option for every item needed. Upon experience and continued exposure with PLDs, many may choose to go with manufacturers not listed in this section. What is suggested here is only meant to be a starting point for those who do not know where to begin.

PLD: The PLD Companion prototyping board was designed to be compatible with a number of PLDs, including ones that are smaller and cheaper than the one listed below. However, for the purposes of this document, we will look at only one. The PLD listed below has 22 inputs and 10 outputs and is electrically erasable, which means it can be reprogrammed many times over.

Product: GAL 22v10
Manufacturer: Lattice
Where: www.jameco.com
Jameco Part #: 39167
Price: \$3.99

Programmer: You will need to buy or have access to a PLD programmer. The one listed below is relatively inexpensive (as far as programmers go), easy to use, and offers a wide range of programming capabilities.

Product: Emp-11
Manufacturer: Needhams Electronics
Where: www.needhams.com
Price: \$334.95

Programming Software: Below is a software package that will be needed to program our PLDs. It is free to download if you register. Type in the web address below, click on “Register to Download”, and follow the instructions from there.

Product: WinCUPL
Manufacturer: Atmel Corporation
Where: <http://www.atmel.com/products/PLD/>

- click on “tools and software”
- click on “WinCUPL”
- click on “Register to Download”

Price: Free

C.The Pins of the PLD

To begin with, it might be helpful to discuss, at some level, what is inside of a PLD and, more specifically, what is inside of the GAL 22v10. The word “GAL” stands for “Generic Array of Logic.” This name is a result of an array of possible connections which can be made within the chip. These connections will logically connect your inputs to your outputs. The “22v10” tells us that this chip has 22 inputs and 10 outputs. Since there are only 24 pins on the chip, we might deduce, and accurately so, that some of the inputs are also outputs. In fact, there are 12 *exclusive* inputs, 0 *exclusive* outputs, and 10 pins that can act as inputs *and* outputs. Figure 1.1 gives a diagram of the general pin assignments on the GAL 22v10.

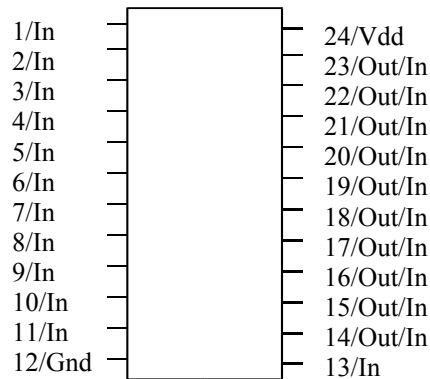


Figure 1.1
Diagram of Pin function on GAL 22v10

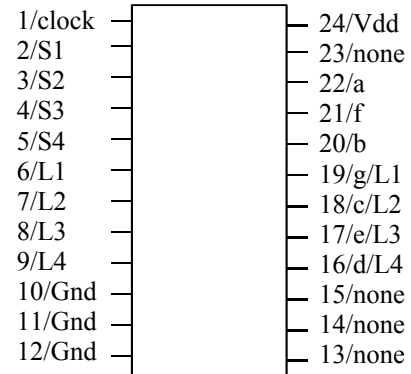


Figure 1.2
Diagram of Electrical Connections on PLD Companion

If you could look inside the chip (technical documents are available at www.jameco.com), you will see that every output pin has a 'D' flip flop associated with it. When designing sequential counters, it is necessary to use these 'Q' outputs as the inputs of our logic. For this reason, it is especially necessary that the outputs wrap back into the circuit internally to be reused as inputs. For this reason and others, all 10 output pins can and will be used as input pins as well.

Figure 1.2 is a diagram of the specific connections that are made between the Breadboard Companion power supply kit and the PLD Companion prototyping board. You will notice that pin 1 is where the clock enters the PLD. Though the clock is itself an input, pin 1 is the only pin that can attach to the clock inputs of the 'D' flip flops. If we are going to have a clock, it must go into pin 1. In the same way, the switches from the power supply are fixed to the input pins 2-5 and the LEDs are fixed to pins 6-9 and pins 16-19. Because the LEDs are primarily intended to be outputs, we will be using pins 16-19 for some of our circuits in which we need four LED

outputs. Though a design possibility, it is not recommended to use the LEDs as additional inputs and for that reason we will never use pins 6-9. Pins 16-23 are fixed to the LED segments of the 7-segment display and their individual assignments are given both on the board and in the diagram. Finally, to comply with smaller and cheaper PLDs, pins 10 and 11 have been grounded in addition to pin 12. For the purposes of this document, however, we will not use pins 10 or 11.

Upon looking at the output pins 16-19, you will notice that they attach to both the 7-segment display *and* the four LEDs of the Breadboard Companion. Unfortunately, we will not be able to split this output into two separate outputs for any given circuit. This means that we must choose, for any given circuit, whether we want to use the 7-segment display *or* the four LEDs of the Companion to be our outputs for that circuit. For this reason, if we choose to use L1-L4 for our outputs, we need to ignore the 7-segment display and if we choose to use the 7-segment display we need to ignore L1-L4. Though it appears confusing at first, it allows for a large number of circuits to be designed and tested with minimal hardware requirements.

D. The Process of Programming a PLD

In this section, we will attempt to outline the process of programming a PLD in as concise and as straightforward a manner as possible. We will assume that the inputs, outputs, and equations for our project are known, we have all of the equipment necessary, and all of our software is installed.

Step 1—WinCUPL:

Once you have obtained the equations for a particular project, the first step will always involve entering them into WinCUPL. Start WinCUPL, then go to “File-->New-->Project”. You will be asked to enter information about the project. Two fields are extremely important: “Name” and “Device”. The “Name” is going to be the name of your file (choose whatever name you like, but try to keep it relatively simple) and the “Device” will be “g22v10”. Upon entering these two fields, hit “OK” and you will be asked for the number of your inputs, the number of your outputs and the number of pinnodes. For the GAL22v10, the number of pinnodes will always be zero. Do not worry if you enter any of these fields incorrectly, you can always change them later on.

Entering all of the initial information gives you a template for writing your program (see Parking Space Problem Example below). The template can be edited by simply retyping the information on the screen. The template is divided up into 4 sections: project info, inputs, outputs, and white space.

Project Info: As long as the “Name” and the “Device” are filled out from the initial prompts, you will not have to edit the project info.

Inputs: You will need to identify your pin numbers for each input you intend on using and assign a variable for each as well (see example below).

Outputs: You will need to identify your pin numbers for each output you intend on using and assign a variable for each as well.

White Space: Below the “output” section of the template you will find white space. The reason why this is referred to as a section of the template is because this is where you write your equations. You should write an equation for every defined output. When writing your equations, use the following syntax.

AND	--> &
OR (+)	--> #
NOT	--> ! (precedes variable)
End of Equation	--> ;

An example of a completed program is given below.

Parking Space Problem Example:

```

Name      Parking ;
PartNo    00 ;
Date      5/15/2005 ;
Revision  01 ;
Designer  Engineer ;
Company   Faith Christian High School ;
Assembly  None ;
Location  ;
Device    g22v10 ;

/* ***** INPUT PINS ***** */
PIN      2 = P1      ; /* */
PIN      3 = P2      ; /* */
PIN      4 = P3      ; /* */
PIN      5 = P4      ; /* */

/* ***** OUTPUT PINS ***** */
PIN      19 = A1     ; /* */
PIN      18 = A2     ; /* */
PIN      17 = A3     ; /* */
PIN      16 = A4     ; /* */

A4 = P4&P3&P2&P1;
A3 = P3&P2&P1 # P4&P3&P2;
A2 = P2&P1 # P3&P2 # P4&P3;
A1 = P1 # P2 # P3 # P4;

```

Step 2—Programming the PLD with the jedec file:

Once the program template has been filled out correctly, it is time to compile the program and obtain a jedec file. Compiling a program allows the software to check and see if it knows what you are asking it to do. It will check and make sure all the syntax makes sense and is in order. A successful

compilation does not mean your circuit will work, it just means the software understands what you are asking it to do. You compile your program by clicking on “Run-->Device Dependent Compile” from the menu bar. If the program compiles, then you should see a jedec file (i.e. Parking.jed) in a window on the right hand side of your screen. If it does not compile, then you need to edit the incorrect syntax, save the changes, and try to compile again.

If your program compiles correctly, but you do not see a jedec file in the window on your right, then it is probably due to one of two problems. First, and most common, check and see if the device is labeled as “g22v10” under the program information. If it has the default “virtual”, then you will not get a jedec file. If this is your problem, then simply make the change and try again. If this is not your problem, go to “Options-->Compiler” from the menu bar, click on the “General” tab and then check the box marked “JEDEC name = PLD name”. Hit “OK” and try the compilation again.

Now that you have created a jedec file (it will probably be under the c:/wincupl/wincupl directory), you will need to import it into the Emp-11 programming software. Place your PLD into the Emp-11 (look at the picture on the programmer to see how) and lock it into place. Make sure your Emp-11 is connected to your computer, then start the Emp-11 programming software. You will need to do the following:

1. Select the Emp-11 programmer
2. Select the PLD:
 - a. Lattice is the manufacturer
 - b. GAL22v10 is the device
 - c. Module M3A will need to be inserted into the Emp-11
3. Erase the PLD
4. Load your jedec file:
 - a. Click on “edit buffer”
 - b. Click “open File” and select the jedec file you want to import
 - c. Exit the buffer editor
5. Click “program”

If, for some reason, the PLD fails to program successfully, check and see if the PLD is oriented correctly in the programmer and repeat the steps outlined above. Once the PLD is programmed, you may insert it into the PLD Companion prototyping board and see if your digital circuit really works.

Step 3—Troubleshooting:

There are many reasons why a programmed PLD may not be working. It might be a hardware problem, a software problem, or a problem in the equations themselves. As with any troubleshooting, the key is to isolate the problem.

Hardware—The hardware is the most likely candidate for a problem if the programmed chip gives no response whatsoever. (1) Make sure the PLD is correctly inserted into the PLD Companion. (2) Try using a different PLD and make sure it is oriented in the Emp-11 correctly while programming. (3) If this is your first time testing a PLD with the PLD Companion prototyping board, test the soldered connections as it suggests in the *Assembly Instructions* for the PLD Companion (available on our website).

Software—This is another possible place for error if there is no response out of the PLD. However, if the PLD is doing something, but nothing close to what was attempted to be programmed, then it might be a software problem. (1) Repeat the programming process, making sure that the program has compiled successfully. (2) Check the modification time of the jedec file and make sure it is new. (3) Change the name of the jedec file (but leave the .jed extension) so that there will be no mix-up with old jedec files.

Equations—Though equations determine the entire function of the PLD, this is the most likely problem if most of the circuit is working, but some of it is not. (1) If the circuit is behaving erratically, re-check the pin numbers and pin assignments, making sure the pin *numbers* are on the left and *assignments* on the right. (2) If only one LED is not behaving as it should, check the equation associated with that particular LED output.

Now that you have the tools needed to successfully program a PLD, it is time to tackle some projects.

2. Combinational Logic Projects—Ideas for BBCII or BBCJr Power Supply Kits

A. Parking Space Problem

Problem: An individual has 4 parking spaces outside of his/her apartment complex. This individual wants to know when two adjacent spaces are open as he/she does not want anyone to park next to their car. (Normally, this person parks at the end of a large parking lot to avoid any scrapes or scratches). This individual proceeds to set up pressure switches in the parking spaces and has a logic indicator in their apartment building. They want one logic indicator to activate if two adjacent spaces become open, another logic indicator to activate if three adjacent spaces become open, a third if all spaces are open and a fourth if one space is open (in case they choose to risk it).

Truth Table:

<i>Inputs</i>					<i>Outputs</i>			
P4	P3	P2	P1		A4	A3	A2	A1
0	0	0	0		0	0	0	0
0	0	0	1		0	0	0	1
0	0	1	0		0	0	0	1
0	0	1	1		0	0	1	1
0	1	0	0		0	0	0	1
0	1	0	1		0	0	0	1
0	1	1	0		0	0	1	1
0	1	1	1		0	1	1	1
1	0	0	0		0	0	0	1
1	0	0	1		0	0	0	1
1	0	1	0		0	0	0	1
1	0	1	1		0	0	1	1
1	1	0	0		0	0	1	1
1	1	0	1		0	0	1	1
1	1	1	0		0	1	1	1
1	1	1	1		1	1	1	1

Inputs:

Parking Space 1 – P1
 Parking Space 2 – P2
 Parking Space 3 – P3
 Parking Space 4 – P4

Outputs:

Only One Space Open – A1
 Two or More Adjacent Spaces Open – A2
 Three or More Adjacent Spaces Open – A3
 Four Adjacent Spaces Open – A4

Note: Outputs will display on the four LEDs of the power supply kit

PLD Pin Numbers:

Inputs:	Outputs:
PIN 2 = P1	PIN 19 = A1
PIN 3 = P2	PIN 18 = A2
PIN 4 = P3	PIN 17 = A3
PIN 5 = P4	PIN 16 = A4

An example of how to enter this project into WinCUPL is given on page 7

Unsimplified Boolean Equations:

$$A4 = P4 \& P3 \& P2 \& P1;$$

$$A3 = !P4 \& P3 \& P2 \& P1 \# P4 \& P3 \& P2 \& !P1 \# P4 \& P3 \& P2 \& P1;$$

$$A2 = !P4 \& !P3 \& P2 \& P1 \# !P4 \& P3 \& P2 \& !P1 \# !P4 \& P3 \& P2 \& P1 \# P4 \& !P3 \& P2 \& P1 \#$$

$$P4 \& P3 \& !P2 \& !P1 \# P4 \& P3 \& !P2 \& P1 \# P4 \& P3 \& P2 \& !P1 \# P4 \& P3 \& P2 \& P1;$$

$$A1 = !P4 \& !P3 \& !P2 \& P1 \# !P4 \& !P3 \& P2 \& !P1 \# !P4 \& !P3 \& P2 \& P1 \# !P4 \& P3 \& !P2 \& !P1 \#$$

$$!P4 \& P3 \& !P2 \& P1 \# !P4 \& P3 \& P2 \& !P1 \# !P4 \& P3 \& P2 \& P1 \# P4 \& !P3 \& !P2 \& !P1 \#$$

$$P4 \& !P3 \& !P2 \& P1 \# P4 \& !P3 \& P2 \& !P1 \# P4 \& !P3 \& P2 \& P1 \# P4 \& P3 \& !P2 \& !P1 \#$$

$$P4 \& P3 \& !P2 \& P1 \# P4 \& P3 \& P2 \& !P1 \# P4 \& P3 \& P2 \& P1;$$

Simplified Boolean Equations:

$$A4 = P4 \& P3 \& P2 \& P1;$$

$$A3 = P3 \& P2 \& P1 \# P4 \& P3 \& P2;$$

$$A2 = P2 \& P1 \# P3 \& P2 \# P4 \& P3;$$

$$A1 = P1 \# P2 \# P3 \# P4;$$

Additional Projects: Think of projects that solve problems. Any circuit with no more than four inputs and four LED outputs can be done in this way. Be creative!

B. Binary to Hexadecimal Decoder

Problem: Design a decoder that translates binary into its hexadecimal equivalent. The binary input will come from the 4 digital switches and the hexadecimal output will be sent to the common anode, 7-segment display.

Truth Table: We are working with a *Common Anode Display*—this means that a “0” in the truth table will turn the LED segment “on”.

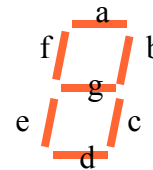
<i>Inputs</i>				<i>Outputs</i>							
S4	S3	S2	S1		a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	0	1	1	1	1
0	0	1	0	2	0	0	1	0	0	1	0
0	0	1	1	3	0	0	0	0	1	1	0
0	1	0	0	4	1	0	0	1	1	0	0
0	1	0	1	5	0	1	0	0	1	0	0
0	1	1	0	6	0	1	0	0	0	0	0
0	1	1	1	7	0	0	0	1	1	1	1
1	0	0	0	8	0	0	0	0	0	0	0
1	0	0	1	9	0	0	0	1	1	0	0
1	0	1	0	A	0	0	0	1	0	0	0
1	0	1	1	b	1	1	0	0	0	0	0
1	1	0	0	C	0	1	1	0	0	0	1
1	1	0	1	d	1	0	0	0	0	1	0
1	1	1	0	E	0	1	1	0	0	0	0
1	1	1	1	F	0	1	1	1	0	0	0

Inputs:

- Least Significant Binary Bit – S1
- Middle Right Binary Bit – S2
- Middle Left Binary Bit – S3
- Most Significant Binary Bit – S4

Outputs:

- a
- b
- c
- d
- e
- f
- g



PLD Pin Numbers:

Inputs:

- PIN 2 = S1
- PIN 3 = S2
- PIN 4 = S3
- PIN 5 = S4

Outputs:

- PIN 22 = a
- PIN 21 = f
- PIN 20 = b
- PIN 19 = g
- PIN 18 = c
- PIN 17 = e
- PIN 16 = d

Unsimplified Boolean Equations:

$$\begin{aligned}
 a &= !S4 \& !S3 \& !S2 \& S1 \# !S4 \& S3 \& !S2 \& !S1 \# S4 \& !S3 \& S2 \& S1 \# S4 \& S3 \& !S2 \& S1; \\
 b &= !S4 \& S3 \& !S2 \& S1 \# !S4 \& S3 \& S2 \& !S1 \# S4 \& !S3 \& S2 \& S1 \# S4 \& S3 \& !S2 \& !S1 \# \\
 &\quad S4 \& S3 \& S2 \& !S1 \# S4 \& S3 \& S2 \& S1; \\
 c &= !S4 \& !S3 \& S2 \& !S1 \# S4 \& S3 \& !S2 \& !S1 \# S4 \& S3 \& S2 \& !S1 \# S4 \& S3 \& S2 \& S1; \\
 d &= !S4 \& !S3 \& !S2 \& S1 \# !S4 \& S3 \& !S2 \& !S1 \# !S4 \& S3 \& S2 \& S1 \# S4 \& !S3 \& !S2 \& S1 \# \\
 &\quad S4 \& !S3 \& S2 \& !S1 \# S4 \& S3 \& S2 \& S1; \\
 e &= !S4 \& !S3 \& !S2 \& S1 \# !S4 \& !S3 \& S2 \& S1 \# !S4 \& S3 \& !S2 \& !S1 \# !S4 \& S3 \& !S2 \& S1 \# \\
 &\quad !S4 \& S3 \& S2 \& S1 \# S4 \& !S3 \& !S2 \& S1; \\
 f &= !S4 \& !S3 \& !S2 \& S1 \# !S4 \& !S3 \& S2 \& !S1 \# !S4 \& !S3 \& S2 \& S1 \# !S4 \& S3 \& S2 \& S1 \# \\
 &\quad S4 \& S3 \& !S2 \& S1; \\
 g &= !S4 \& !S3 \& !S2 \& !S1 \# !S4 \& !S3 \& !S2 \& S1 \# !S4 \& S3 \& S2 \& S1 \# S4 \& S3 \& !S2 \& !S1;
 \end{aligned}$$

Simplified Boolean Equations:

$$\begin{aligned}
 a &= !S4 \& !S3 \& !S2 \& S1 \# !S4 \& S3 \& !S2 \& !S1 \# S4 \& !S3 \& S2 \& S1 \# S4 \& S3 \& !S2 \& S1; \\
 b &= !S4 \& S3 \& !S2 \& S1 \# S3 \& S2 \& !S1 \# S4 \& S3 \& !S1 \# S4 \& S2 \& S1; \\
 c &= !S4 \& !S3 \& S2 \& !S1 \# S4 \& S3 \& !S1 \# S4 \& S3 \& S2; \\
 d &= !S3 \& !S2 \& S1 \# !S4 \& S3 \& !S2 \& !S1 \# S3 \& S2 \& S1 \# S4 \& !S3 \& S2 \& !S1; \\
 e &= !S4 \& S1 \# !S4 \& S3 \& !S2 \# !S3 \& !S2 \& S1; \\
 f &= !S4 \& !S3 \& S1 \# !S4 \& !S3 \& S2 \# !S4 \& S2 \& S1 \# S4 \& S3 \& !S2 \& S1; \\
 g &= !S4 \& !S3 \& !S2 \# !S4 \& S3 \& S2 \& S1 \# S4 \& S3 \& !S2 \& !S1;
 \end{aligned}$$

Additional Projects: Try building many different types of “decoders”. Try counting down in binary as the inputs count up; try to count by 2's in binary as the inputs count by 1's; count out the date, or any other sequence of numbers that you like.

C.Sixteen Character Phrase

Problem: Design a circuit that displays a phrase of your choice on the 7-segment display. There will be 16 characters in the phrase and their order will be dictated by the binary input of the four digital switches.

Truth Table: We are working with a *Common Anode Display*—this means that a “0” in the truth table will turn the LED segment “on”. The phrase chosen for this project is “good bad and ugly”. In order for it to fit into 16 states, we had to cut out the last space.

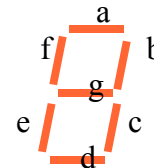
<i>Inputs</i>				<i>Outputs</i>							
S4	S3	S2	S1		a	b	c	d	e	f	g
0	0	0	0	g	0	0	0	0	1	0	0
0	0	0	1	o	1	1	0	0	0	1	0
0	0	1	0	o	1	1	0	0	0	1	0
0	0	1	1	d	1	0	0	0	0	1	0
0	1	0	0		1	1	1	1	1	1	1
0	1	0	1	b	1	1	0	0	0	0	0
0	1	1	0	A	0	0	0	1	0	0	0
0	1	1	1	d	1	0	0	0	0	1	0
1	0	0	0		1	1	1	1	1	1	1
1	0	0	1	A	0	0	0	1	0	0	0
1	0	1	0	n	1	1	0	1	0	1	0
1	0	1	1	d	1	0	0	0	0	1	0
1	1	0	0	U	1	0	0	0	0	0	1
1	1	0	1	g	0	0	0	0	1	0	0
1	1	1	0	L	1	1	1	0	0	0	1
1	1	1	1	y	1	0	0	0	1	0	0

Inputs:

Least Significant Binary Bit – S1
 Middle Right Binary Bit – S2
 Middle Left Binary Bit – S3
 Most Significant Binary Bit – S4

Outputs:

a
b
c
d
e
f
g



PLD Pin Numbers:

Inputs:

PIN 2 = S1
 PIN 3 = S2
 PIN 4 = S3
 PIN 5 = S4

Outputs:

PIN 22 = a
 PIN 21 = f
 PIN 20 = b
 PIN 19 = g
 PIN 18 = c
 PIN 17 = e
 PIN 16 = d

Unsimplified Boolean Equations:

$a = !S4 \& !S3 \& !S2 \& S1 \# !S4 \& !S3 \& S2 \& !S1 \# !S4 \& !S3 \& S2 \& S1 \# !S4 \& S3 \& !S2 \& !S1 \#$
 $!S4 \& S3 \& !S2 \& S1 \# !S4 \& S3 \& S2 \& S1 \# S4 \& !S3 \& !S2 \& !S1 \# S4 \& !S3 \& S2 \& !S1 \#$
 $S4 \& !S3 \& S2 \& S1 \# S4 \& S3 \& !S2 \& !S1 \# S4 \& S3 \& S2 \& !S1 \# S4 \& S3 \& S2 \& S1;$
 $b = !S4 \& !S3 \& !S2 \& S1 \# !S4 \& !S3 \& S2 \& !S1 \# !S4 \& S3 \& !S2 \& !S1 \# !S4 \& S3 \& !S2 \& S1 \#$
 $S4 \& !S3 \& !S2 \& !S1 \# S4 \& !S3 \& S2 \& !S1 \# S4 \& S3 \& S2 \& !S1;$
 $c = !S4 \& S3 \& !S2 \& !S1 \# S4 \& !S3 \& !S2 \& !S1 \# S4 \& S3 \& S2 \& !S1;$
 $d = !S4 \& S3 \& !S2 \& !S1 \# !S4 \& S3 \& S2 \& !S1 \# S4 \& !S3 \& !S2 \& !S1 \# S4 \& !S3 \& !S2 \& S1 \#$
 $S4 \& !S3 \& S2 \& !S1;$
 $e = !S4 \& !S3 \& !S2 \& !S1 \# !S4 \& S3 \& !S2 \& !S1 \# S4 \& !S3 \& !S2 \& !S1 \# S4 \& S3 \& !S2 \& S1 \#$
 $S4 \& S3 \& S2 \& S1;$
 $f = !S4 \& !S3 \& !S2 \& S1 \# !S4 \& !S3 \& S2 \& !S1 \# !S4 \& !S3 \& S2 \& S1 \# !S4 \& S3 \& !S2 \& !S1 \#$
 $!S4 \& S3 \& S2 \& S1 \# S4 \& !S3 \& !S2 \& !S1 \# S4 \& !S3 \& S2 \& !S1 \# S4 \& !S3 \& S2 \& S1;$
 $g = !S4 \& S3 \& !S2 \& !S1 \# S4 \& !S3 \& !S2 \& !S1 \# S4 \& S3 \& !S2 \& !S1 \# S4 \& S3 \& S2 \& !S1;$

Simplified Boolean Equations:

$a = S2 \& S1 \# !S4 \& S1 \# S4 \& !S1 \# !S4 \& !S3 \& S2 \# S3 \& !S2 \& !S1;$
 $b = !S4 \& S3 \& !S2 \# !S4 \& !S2 \& S1 \# S4 \& S2 \& !S1 \# S4 \& !S3 \& !S1 \# !S3 \& S2 \& !S1;$
 $c = !S4 \& S3 \& !S2 \& !S1 \# S4 \& !S3 \& !S2 \& !S1 \# S4 \& S3 \& S2 \& !S1;$
 $d = S4 \& !S3 \& !S2 \# !S4 \& S3 \& !S1 \# S4 \& !S3 \& !S1;$
 $e = !S4 \& !S2 \& !S1 \# S4 \& S3 \& S1 \# !S3 \& !S2 \& !S1;$
 $f = S4 \& !S3 \& S2 \# S4 \& !S3 \& S1 \# !S4 \& S2 \& S1 \# !S4 \& !S3 \& S1 \# !S4 \& !S3 \& S2 \# !$
 $S4 \& S3 \& !S2 \& !S1;$
 $g = !S4 \& S3 \& !S2 \& !S1 \# S4 \& !S3 \& !S2 \& !S1 \# S4 \& S3 \& !S2 \& !S1 \# S4 \& S3 \& S2 \& !S1;$

Additional Projects: Any phrase that can be written on a 7-segment display can be done. Keep in mind that some letters do not work so well or not at all (i.e. “x”). The challenge comes in finding a meaningful phrase that uses only the letters available and is the appropriate number of characters in length. This project can be fun as it gives the designer a chance to personalize their circuit.

3. Sequential Logic Projects—Ideas for the BBCII Power Supply Kit

A. 0000-1111 Binary Counter

Problem: Design a circuit that automatically counts in Binary from 0000 to 1111 on the four logic indicators of the Breadboard Companion power supply kit. The frequency of the clock pulse will dictate the speed of this counting circuit.

Truth Table: The “Q” output of the D-flip flops will be what drives the lights that count in binary. However, these “Q” outputs will also wrap back into the circuit and feed the logic we need to determine the next state. For this reason, we will identify the input of our logic as “Q” and the output of our logic as “D”. We will identify “Q” as representing the present state of our counter and “D” as representing the next state. We are writing equations for “D” because we want to write equations *for* the next state and *in terms of* the present state.

<i>Now State</i>					<i>Next State</i>			
Q4	Q3	Q2	Q1		D4	D3	D2	D1
0	0	0	0		0	0	0	1
0	0	0	1		0	0	1	0
0	0	1	0		0	0	1	1
0	0	1	1		0	1	0	0
0	1	0	0		0	1	0	1
0	1	0	1		0	1	1	0
0	1	1	0		0	1	1	1
0	1	1	1		1	0	0	0
1	0	0	0		1	0	0	1
1	0	0	1		1	0	1	0
1	0	1	0		1	0	1	1
1	0	1	1		1	1	0	0
1	1	0	0		1	1	0	1
1	1	0	1		1	1	1	0
1	1	1	0		1	1	1	1
1	1	1	1		0	0	0	0

Inputs:

Clock pulse

Outputs:

Least Significant Bit – Q1

Middle Right Bit – Q2

Middle Left Bit – Q3

Most Significant Bit – Q4

Note: Outputs will display on the four LEDs of the power supply kit

PLD Pin Numbers:

Inputs:

PIN 1 = clock

Outputs:

PIN 19 = Q1

PIN 18 = Q2

PIN 17 = Q3

PIN 16 = Q4

Unsimplified Boolean Equations:

$$\begin{aligned}
 Q4.d &= !Q4 \& Q3 \& Q2 \& Q1 \# Q4 \& !Q3 \& !Q2 \& !Q1 \# Q4 \& !Q3 \& !Q2 \& Q1 \# Q4 \& !Q3 \& Q2 \& !Q1 \# \\
 & Q4 \& !Q3 \& Q2 \& Q1 \# Q4 \& Q3 \& !Q2 \& !Q1 \# Q4 \& Q3 \& !Q2 \& Q1 \# Q4 \& Q3 \& Q2 \& !Q1; \\
 Q3.d &= !Q4 \& !Q3 \& Q2 \& Q1 \# !Q4 \& Q3 \& !Q2 \& !Q1 \# !Q4 \& Q3 \& !Q2 \& Q1 \# !Q4 \& Q3 \& Q2 \& !Q1 \# \\
 & Q4 \& !Q3 \& Q2 \& Q1 \# Q4 \& Q3 \& !Q2 \& !Q1 \# Q4 \& Q3 \& !Q2 \& Q1 \# Q4 \& Q3 \& Q2 \& !Q1; \\
 Q2.d &= !Q4 \& !Q3 \& !Q2 \& Q1 \# !Q4 \& !Q3 \& Q2 \& !Q1 \# !Q4 \& Q3 \& !Q2 \& Q1 \# !Q4 \& Q3 \& Q2 \& !Q1 \# \\
 & Q4 \& !Q3 \& !Q2 \& Q1 \# Q4 \& !Q3 \& Q2 \& !Q1 \# Q4 \& Q3 \& !Q2 \& Q1 \# Q4 \& Q3 \& Q2 \& !Q1; \\
 Q1.d &= !Q4 \& !Q3 \& !Q2 \& !Q1 \# !Q4 \& !Q3 \& Q2 \& !Q1 \# !Q4 \& Q3 \& !Q2 \& !Q1 \# !Q4 \& Q3 \& Q2 \& !Q1 \# \\
 & Q4 \& !Q3 \& !Q2 \& !Q1 \# Q4 \& !Q3 \& Q2 \& !Q1 \# Q4 \& Q3 \& !Q2 \& !Q1 \# Q4 \& Q3 \& Q2 \& !Q1;
 \end{aligned}$$

Simplified Boolean Equations:

$$\begin{aligned}
 Q4.d &= !Q4 \& Q3 \& Q2 \& Q1 \# Q4 \& !Q3 \# Q4 \& !Q2 \# Q4 \& !Q1; \\
 Q3.d &= !Q3 \& Q2 \& Q1 \# Q3 \& !Q2 \# Q3 \& !Q1; \\
 Q2.d &= Q2 \& !Q1 \# !Q2 \& Q1; \\
 Q1.d &= !Q1;
 \end{aligned}$$

Additional Projects: Sequential circuits are fun because they are “automated” and because they can be designed to count in any order. They are identical to combinational logic circuits except no one has to flip any switches. Try designing circuits that count down in binary, count odd numbers in binary, count out your phone number in binary (as long as there are no repetitive digits), or just have the lights flash in some sequential order.

B. Automated Phrase or Numeric Sequence

Problem: Though phrases and numeric combinations abound, we will design a circuit that automatically counts down in Decimal on the 7-segment Display from “8” to “0” and then repeat.

Truth Table: This circuit will be similar to the last in which we will write equations for the “next state” and make them depend on the “present state”. The only difference being the number and type of output LEDs we are working with. Remember, we are working with a *Common Anode Display*—this means that a “0” in the truth table will turn the LED segment “on”.

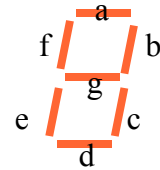
	<i>Now State (q)</i>							<i>Next State (d)</i>							
	a	b	c	d	e	f	g		a	b	c	d	e	f	g
8	0	0	0	0	0	0	0	7	0	0	0	1	1	1	1
7	0	0	0	1	1	1	1	6	0	1	0	0	0	0	0
6	0	1	0	0	0	0	0	5	0	1	0	0	1	0	0
5	0	1	0	0	1	0	0	4	1	0	0	1	1	0	0
4	1	0	0	1	1	0	0	3	0	0	0	0	1	1	0
3	0	0	0	0	1	1	0	2	0	0	1	0	0	1	0
2	0	0	1	0	0	1	0	1	1	0	0	1	1	1	1
1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	8	0	0	0	0	0	0	0

Inputs:

Clock pulse

Outputs:

a
b
c
d
e
f
g



PLD Pin Numbers:

Inputs:

PIN 1 = clock

Outputs:

PIN 22 = a
PIN 21 = f
PIN 20 = b
PIN 19 = g
PIN 18 = c
PIN 17 = e
PIN 16 = d

Unsimplified Boolean Equations:

$a.d = !a \& !b \& !c \& !d \& !e \& !f \& !g \# !a \& !b \& c \& !d \& !e \& f \& !g;$
 $b.d = !a \& !b \& !c \& d \& !e \& f \& !g \# !a \& b \& !c \& !d \& !e \& !f \& !g;$
 $c.d = !a \& !b \& !c \& !d \& !e \& f \& !g;$
 $d.d = !a \& !b \& !c \& !d \& !e \& !f \& !g \# !a \& b \& !c \& !d \& !e \& !f \& !g \# !a \& !b \& c \& !d \& !e \& f \& !g;$
 $e.d = !a \& !b \& !c \& !d \& !e \& !f \& !g \# !a \& b \& !c \& !d \& !e \& !f \& !g \# !a \& b \& !c \& !d \& !e \& !f \& !g \# !a \& !b \& !c \& d \& !e \& !f \& !g \# !a \& !b \& c \& !d \& !e \& !f \& !g;$
 $f.d = !a \& !b \& !c \& !d \& !e \& !f \& !g \# a \& !b \& !c \& d \& !e \& !f \& !g \# !a \& !b \& !c \& !d \& !e \& f \& !g \# !a \& !b \& c \& !d \& !e \& !f \& !g;$
 $g.d = !a \& !b \& !c \& !d \& !e \& !f \& !g \# !a \& !b \& c \& !d \& !e \& !f \& !g \# a \& !b \& !c \& d \& !e \& !f \& !g;$

Simplified Boolean Equations:

Simplification is insignificant.

Additional Projects: Using a method similar to this one, the number and types of circuits possible are virtually boundless. Any phrase of letters or numbers can be designed to display in an automated sequence on the 7-segment display. However, there are a few limitations to consider.

- Theoretically, 2^7 characters can be displayed in any one sequence. However, you will run out of AND gates on the PLD before you reach this long of a phrase. If using the GAL 22v10, the equations that your phrase requires cannot exceed 10 AND gates for the a LED segment, 12 AND gates for the f and d segments, 14 AND gates for the b and e segments, and 16 AND gates for the g and c segments.
- Each character that you use in your phrase or numeric sequence must be unique. This is because the next output depends on the present one, and thus no two states (characters) can be identical.
- Third, because all of the output pins will default to “0” when first powered up, all phrases must begin with the number “8”. However, though they must begin with this number, they do not have to return to it. Aside from the initial run, “8” does not have to be in the repetitive sequence.

Possible circuits might include the date of your birth, your street address, a phrase of your choice, or your “mother's maiden name.” This project is wide open for creativity.

4. Final Project—Idea for the BBCII Power Supply Kit

Sequential/Combinational Jamboree

Problem: Design a circuit that runs through four different phrases, depending which switch on the dip switch is activated. The circuit will include four combinational logic circuits and one 3-bit counter. The 3-bit counter will drive the phrases. The activation of each phrase will depend on the activation of one of the four digital switches. This project will involve five sub-projects and will require a PLD with 10 outputs.

- I. *3-bit Counter*—The first circuit we will build is a 3-bit counter. This will be the counter that will drive our combinational logic circuits. This counter is kind of like an invisible hand flipping through the switches of a combinational logic circuit. We will build this counter on output pins 23, 14, and 15 of our PLD so that the pins that drive the 7-segment display will remain free.

Truth Table: Refer to 3A of this hand-out as a reminder of how to build a sequential, binary counter. For the sake of simplicity, we will only list the *simplified* boolean equations.

<i>Now State</i>				<i>Next State</i>		
Q3	Q2	Q1		D3	D2	D1
0	0	0		0	0	1
0	0	1		0	1	0
0	1	0		0	1	1
0	1	1		1	0	0
1	0	0		1	0	1
1	0	1		1	1	0
1	1	0		1	1	1
1	1	1		0	0	0

PLD Pin Numbers:

Inputs:

PIN 1 = clock

Outputs:

PIN 23 = Q3

PIN 15 = Q2

PIN 14 = Q1

Simplified Boolean Equations:

$$Q3.d = Q3 \& !Q2 \# !Q3 \& Q2 \& Q1 \# Q3 \& Q2 \& !Q1;$$

$$Q2.d = Q1 \& !Q2 \# !Q1 \& Q2;$$

$$Q1.d = !Q1;$$

II. *First Phrase*—Though any phrase with eight states will work, we will choose to count 0-7 in decimal on the 7-segment display. We must use eight states as our 3-bit counter is flipping the switches for us and it is already programmed. Since the pre-programmed counter is “flipping our switches”, Q1-Q3 will act as our inputs. Also, we plan on implementing other phrases into this circuit, so we need to make this first phrase dependent on the flip of a manual switch (S1).

Truth Table: For the sake of simplicity, we will only list the *simplified* boolean equations.

<i>Inputs</i>				<i>Outputs</i>							
S1	Q3	Q2	Q1		a	b	c	d	e	f	g
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	1	1	0	1	1	1	1	1
1	0	1	0	2	0	0	0	0	0	1	0
1	0	1	1	3	0	0	0	0	1	1	0
1	1	0	0	4	1	0	0	1	1	0	0
1	1	0	1	5	0	1	0	0	1	0	0
1	1	1	0	6	0	1	0	0	0	0	0
1	1	1	1	7	0	0	0	1	1	1	1

PLD Pin Numbers:

Inputs:

PIN 2 = S1
 PIN 23 = Q3
 PIN 15 = Q2
 PIN 14 = Q1

Outputs:

PIN 22 = a
 PIN 21 = f
 PIN 20 = b
 PIN 19 = g
 PIN 18 = c
 PIN 17 = e
 PIN 16 = d

Simplified Boolean Equations:

a = $S1 \& !Q3 \& !Q2 \& Q1 \# S1 \& Q3 \& !Q2 \& !Q1$;
 b = $S1 \& Q3 \& !Q2 \& Q1 \# S1 \& Q3 \& Q2 \& !Q1$;
 c = $S1 \& !Q3 \& Q2 \& !Q1$;
 d = $S1 \& !Q3 \& !Q2 \& Q1 \# S1 \& Q3 \& !Q2 \& !Q1 \# S1 \& Q3 \& Q2 \& Q1$;
 e = $S1 \& Q1 \# S1 \& Q3 \& !Q2$;
 f = $S1 \& Q2 \& Q1 \# S1 \& !Q3 \& Q1 \# S1 \& !Q3 \& Q2$;
 g = $S1 \& !Q3 \& !Q2 \# S1 \& Q3 \& Q2 \& Q1$;

Note: This circuit will not work without the counter in Part I. If you want to implement these two circuits together only, skip to Part VI to find out how. If you want to implement the entire project, go on to Part III.

III. *Second Phrase*—Though any phrase with eight states will work, we will choose to count down from seven to zero in Decimal on the 7-segment display. Like the last circuit, we will use the 3-bit counter to drive our combinational logic. To keep this combinational circuit distinct from the other ones, we will make it dependent on the flip of a manual switch (S2).

Truth Table: For the sake of simplicity, we will only list the *simplified* boolean equations.

<i>Inputs</i>				<i>Outputs</i>							
S2	Q3	Q2	Q1		a	b	c	d	e	f	g
1	0	0	0	7	0	0	0	1	1	1	1
1	0	0	1	6	0	1	0	0	0	0	0
1	0	1	0	5	0	1	0	0	1	0	0
1	0	1	1	4	1	0	0	1	1	0	0
1	1	0	0	3	0	0	0	0	1	1	0
1	1	0	1	2	0	1	1	0	0	1	0
1	1	1	0	1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	1

PLD Pin Numbers:

Inputs:

PIN 3 = S2
 PIN 23 = Q3
 PIN 15 = Q2
 PIN 14 = Q1

Outputs:

PIN 22 = a
 PIN 21 = f
 PIN 20 = b
 PIN 19 = g
 PIN 18 = c
 PIN 17 = e
 PIN 16 = d

Simplified Boolean Equations:

$a = S2 \& !Q3 \& Q2 \& Q1 \# S2 \& Q3 \& Q2 \& !Q1;$
 $b = S2 \& !Q3 \& !Q2 \& Q1 \# S2 \& !Q3 \& Q2 \& !Q1;$
 $c = S2 \& Q3 \& !Q2 \& Q1;$
 $d = S2 \& !Q3 \& !Q2 \& !Q1 \# S2 \& !Q3 \& Q2 \& Q1 \# S2 \& Q3 \& Q2 \& !Q1;$
 $e = S2 \& !Q1 \# S2 \& !Q3 \& Q2;$
 $f = S2 \& !Q2 \& !Q1 \# S2 \& Q3 \& !Q1 \# S2 \& Q3 \& !Q2;$
 $g = S2 \& Q3 \& Q2 \# S2 \& !Q3 \& !Q2 \& !Q1;$

Note: This circuit will not work without the counter in Part I. If you want to implement some combination of the first three circuits together, skip to Part VI of this project to find out how. If you want to implement the entire project, go on to Part IV.

IV. *Third Phrase*—Though any phrase with eight states will work, we will choose the phrase “go EagLES” to display on the 7-segment display. Because we need eight states, we will skip the space between the words. Like the last circuit, we will use the 3-bit counter to drive our combinational logic. To keep this combinational circuit distinct from the other ones, we will make it dependent on the flip of a manual switch (S3).

Truth Table: For the sake of simplicity, we will only list the *simplified* boolean equations.

<i>Inputs</i>				<i>Outputs</i>							
S3	Q3	Q2	Q1		a	b	c	d	e	f	g
1	0	0	0	g	0	0	0	0	1	0	0
1	0	0	1	o	1	1	0	0	0	1	0
1	0	1	0	E	0	1	1	0	0	0	0
1	0	1	1	A	0	0	0	1	0	0	0
1	1	0	0	g	0	0	0	0	1	0	0
1	1	0	1	L	1	1	1	0	0	0	1
1	1	1	0	E	0	1	1	0	0	0	0
1	1	1	1	S	0	1	0	0	1	0	0

PLD Pin Numbers:

Inputs:

PIN 4 = S3
 PIN 23 = Q3
 PIN 15 = Q2
 PIN 14 = Q1

Outputs:

PIN 22 = a
 PIN 21 = f
 PIN 20 = b
 PIN 19 = g
 PIN 18 = c
 PIN 17 = e
 PIN 16 = d

Simplified Boolean Equations:

a = $S3 \& !Q2 \& Q1$;
 b = $S3 \& !Q2 \& Q1 \# S3 \& Q2 \& !Q1 \# S3 \& Q3 \& Q2$;
 c = $S3 \& Q3 \& !Q2 \& Q1 \# S3 \& Q2 \& !Q1$;
 d = $S3 \& !Q3 \& Q2 \& Q1$;
 e = $S3 \& Q3 \& Q2 \& Q1 \# S3 \& !Q2 \& !Q1$;
 f = $S3 \& !Q3 \& !Q2 \& Q1$;
 g = $S3 \& Q3 \& !Q2 \& Q1$;

Note: This circuit will not work without the counter in Part I. If you want to implement some combination of the first four circuits together, skip to Part VI of this project to find out how. If you want to implement the entire project, go on to Part V.

V. *Fourth Phrase*—Though any phrase with eight states will work, we will choose the phrase “JOHN COOL” to display on the 7-segment display. Because we need eight states, we will skip the space between the words. Like the last circuit, we will use the 3-bit counter to drive our combinational logic. To keep this combinational circuit distinct from the other ones, we will make it dependent on the flip of a manual switch (S4).

Truth Table: For the sake of simplicity, we will only list the *simplified* boolean equations.

<i>Inputs</i>				<i>Outputs</i>							
S4	Q3	Q2	Q1		a	b	c	d	e	f	g
1	0	0	0	J	1	0	0	0	0	1	1
1	0	0	1	O	0	0	0	0	0	0	1
1	0	1	0	H	1	0	0	1	0	0	0
1	0	1	1	n	0	0	0	1	0	0	1
1	1	0	0	C	0	1	1	0	0	0	1
1	1	0	1	O	0	0	0	0	0	0	1
1	1	1	0	O	0	0	0	0	0	0	1
1	1	1	1	L	1	1	1	0	0	0	1

PLD Pin Numbers:

Inputs:

PIN 5 = S4
 PIN 23 = Q3
 PIN 15 = Q2
 PIN 14 = Q1

Outputs:

PIN 22 = a
 PIN 21 = f
 PIN 20 = b
 PIN 19 = g
 PIN 18 = c
 PIN 17 = e
 PIN 16 = d

Simplified Boolean Equations:

a = $S4 \& !Q3 \& !Q1 \# S4 \& Q3 \& Q2 \& Q1$;
 b = $S4 \& Q3 \& !Q2 \& !Q1 \# S4 \& Q3 \& Q2 \& Q1$;
 c = $S4 \& Q3 \& !Q2 \& !Q1 \# S4 \& Q3 \& Q2 \& Q1$;
 d = $S4 \& !Q3 \& Q2$;
 e = 0;
 f = $S4 \& !Q3 \& !Q2 \& !Q1$;
 g = $S4 \& Q3 \# S4 \& !Q2 \# S4 \& Q1$;

Note: This circuit will not work without the counter in Part I. Go on to Part VI to find out how to implement all or some of these circuits together.

VI. *Final Implementation*—Listed Below are the final pin numbers you will need and final, simplified equations to make the complete project work. Each equation for the LED segments are really four sub-equations. Each sub-equation of each equation is “enabled” by the switch that goes with its corresponding AND gate(s). If you want to implement only two combinational logic circuits, then only use S1 and S2 and delete all of the gates that have S3 and S4 in them.

PLD Pin Numbers:

Inputs:

PIN 1 = clock
 PIN 2 = S1
 PIN 3 = S2
 PIN 4 = S3
 PIN 5 = S4

Outputs:

PIN 23 = Q3 PIN 18 = c
 PIN 22 = a PIN 17 = e
 PIN 21 = f PIN 16 = d
 PIN 20 = b PIN 15 = Q2
 PIN 19 = g PIN 14 = Q1

Simplified Boolean Equations:

$$Q3.d = Q3 \& !Q2 \# !Q3 \& Q2 \& Q1 \# Q3 \& Q2 \& !Q1;$$

$$Q2.d = Q1 \& !Q2 \# !Q1 \& Q2;$$

$$Q1.d = !Q1;$$

$$a = S1 \& !Q3 \& !Q2 \& Q1 \# S1 \& Q3 \& !Q2 \& !Q1 \# S2 \& !Q3 \& Q2 \& Q1 \# S2 \& Q3 \& Q2 \& !Q1 \# S3 \& !Q2 \& Q1 \# S4 \& !Q3 \& !Q1 \# S4 \& Q3 \& Q2 \& Q1;$$

$$b = S1 \& Q3 \& !Q2 \& Q1 \# S1 \& Q3 \& Q2 \& !Q1 \# S2 \& !Q3 \& !Q2 \& Q1 \# S2 \& !Q3 \& Q2 \& !Q1 \# S3 \& !Q2 \& Q1 \# S3 \& Q2 \& !Q1 \# S3 \& Q3 \& Q2 \# S4 \& Q3 \& !Q2 \& !Q1 \# S4 \& Q3 \& Q2 \& Q1;$$

$$c = S1 \& !Q3 \& Q2 \& !Q1 \# S2 \& Q3 \& !Q2 \& Q1 \# S3 \& Q3 \& !Q2 \& Q1 \# S3 \& Q2 \& !Q1 \# S4 \& Q3 \& !Q2 \& !Q1 \# S4 \& Q3 \& Q2 \& Q1;$$

$$d = S1 \& !Q3 \& !Q2 \& Q1 \# S1 \& Q3 \& !Q2 \& !Q1 \# S1 \& Q3 \& Q2 \& Q1 \# S2 \& !Q3 \& !Q2 \& !Q1 \# S2 \& !Q3 \& Q2 \& Q1 \# S2 \& Q3 \& Q2 \& !Q1 \# S3 \& !Q3 \& Q2 \& Q1 \# S4 \& !Q3 \& Q2;$$

$$e = S1 \& Q1 \# S1 \& Q3 \& !Q2 \# S2 \& !Q1 \# S2 \& !Q3 \& Q2 \# S3 \& Q3 \& Q2 \& Q1 \# S3 \& !Q2 \& !Q1;$$

$$f = S1 \& Q2 \& Q1 \# S1 \& !Q3 \& Q1 \# S1 \& !Q3 \& Q2 \# S2 \& !Q2 \& !Q1 \# S2 \& Q3 \& !Q1 \# S2 \& Q3 \& !Q2 \# S3 \& !Q3 \& !Q2 \& Q1 \# S4 \& !Q3 \& !Q2 \& !Q1;$$

$$g = S1 \& !Q3 \& !Q2 \# S1 \& Q3 \& Q2 \& Q1 \# S2 \& Q3 \& Q2 \# S2 \& !Q3 \& !Q2 \& !Q1 \# S3 \& Q3 \& !Q2 \& Q1 \# S4 \& Q3 \# S4 \& !Q2 \# S4 \& Q1;$$

Note: If no switches are activated, then an “8” will show up on the display. This is because all of the output pins default to a low when none of the switches are pressed. To get rid of this, include the AND gate $!S1 \& !S2 \& !S3 \& !S4$ with every LED equation. This will cause the output pins to go to a high when no switches are pressed, thus turning off every LED segment.

Additional Projects: It might be possible to include five or more combinational logic circuits into this design. The limiting factor will be in the number of AND gates your simplified circuits require. The number of AND gates on a PLD is not infinite and is fixed depending on which output pin we are talking about. Pin 22 (LED segment *a*) has the fewest number of AND gates at 10 and pins 21 and 16 (*f* and *d*) are a close second at 12.

If you want to add more than 4 combinational logic circuits to your overall project, then you would have to include a two digit switch combination with each AND gate. For example, your first combinational logic circuit may require S1 to be on and S2 to be off; therefore, $\overline{S2} \& S1$ would have to be included in each AND gate for that particular circuit. If every AND gate included a two bit “enabling” across four possible switches, this would make distinct 24 combinational logic circuits.

It might be obvious at this point, but still worth stating, the number and diversity of projects available with nothing more than the Breadboard Companion, PLD Companion, and a PLD. It is quite possible, perhaps even preferable, to learn the basics of designing digital electronic circuits with a very minimum amount of hardware. Because most introductory combinational and sequential logic circuits are quite feasible on a standard PLD, the majority of the design industry works with programmable logic, and because PLD prices continue to decrease, it becomes questionable whether traditional breadboarding with discrete components is worth the time and frustration that often accompanies it. Either way, the PLD Companion was designed to make the field of digital electronics increasingly accessible for those who are interested.