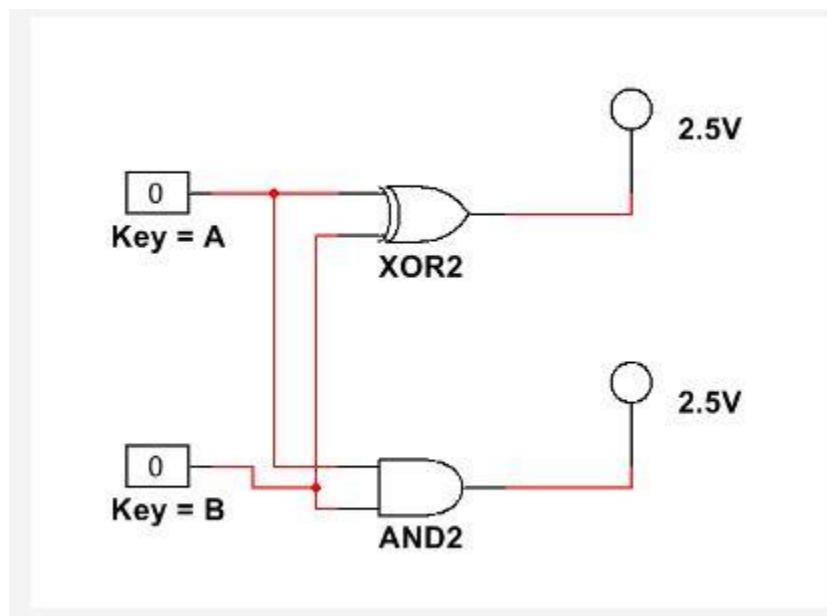


Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 4: Binary Conversion and Adders

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 4: Binary Conversion and Adders

In the first lab, we explored truth tables with two inputs and learned how to design their corresponding circuits. Two binary inputs, namely, 1 and 0 are the simplest of circuits. More complex circuits have more combinations of binary numbers. This makes it impractical to create truth tables for all of the possible combinations and permutations. Instead of creating truth tables, we look to a system that converts our binary numbers to a *binary-coded decimal (BCD)*. Depending on the sequence and each number's position in it, they are assigned a value. Adding up these values gives us the BCD.

Learning Objectives

In this lab, students will:

1. Construct half and full adders with logic gates and create truth tables from them
2. Connect multiple full adders with each other to explore the ripple effect
3. Confirm the truth table for a full adder.

Required Tools and Technology

Platform: NI ELVIS III

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Software: NI LabVIEW FPGA Vivado 2014.4

- ✓ Install:
<http://www.ni.com/download/labview-fpga-module-2015-sp1/5920/en/>

Note: Digilent Driver (The installer above automatically downloads the installer below onto your computer)

- ✓ Navigate to:
C:\NIFPGA\programs\Vivado2014_4\data\xicom\cable_drivers\nt64\digilent
- ✓ Install: install_digilent.exe

Expected Deliverables

In this lab, you will collect the following deliverables:

- Long answer questions regarding adders
- 3 Truth Tables
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

Binary-Coded Decimal

Value	8	4	2	1
Bit	0	0	1	1
Result	0	0	2	1
$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$ Sum = 0 + 0 + 2 + 1 = 3				

Each “bit” is assigned a value

Multiply the value by the bit

Add the results together

The sum is the **binary-coded decimal** value

Figure 1-1 Video. View the video here: <https://youtu.be/YYGAPRracIY>



Video Summary

- In larger circuits it is not practical to make truth tables so binary is converted into a binary-coded decimal (BDC)
- Half adders can be represented in Multisim by a single component or by creating them using an AND and an XOR gate
- Half adders have two inputs and two outputs
- Full adders can be represented in Multisim by a single component or two AND gates and three XOR gates
- Full adders have three inputs and two outputs

Binary-Coded Decimals

Let's look at the following example of a 4-bit binary number (four binary numbers code to one decimal number):

0101

The number furthest to the right is given the value of 1 (similar to the ones column in regular addition). The position immediately to its left is given the value of 2 (similar to a

tens column). The value to the left of this is 4 (similar to the hundreds column). The pattern you will notice is that the values assigned to the number on the left increase by twice the value of the number before it.

In our example, from left to right, the BCD value for the four-bit binary code above, adding up the values is:

$$0 + 4 + 0 + 1 = 5$$

Conversions that result in a single digit number (0-9) are already in BCD format. In order to output the decimal value of a binary number greater than 9 in circuitry, a Binary-to-BCD converter such as a Shift-Add-3 algorithm must be used.

Adders

Half-Adders:

- A half-adder does binary addition on two inputs (A and B).
- The two outputs are labeled sum (S) and carry (C).
- Half adders can be built with:
 - an XOR gate and an AND gate (shown on the left).
 - a component in Multisim (shown on the right).

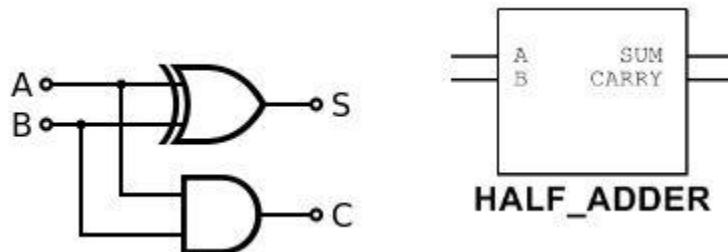


Figure 1-2 Half Adder from Gates

Figure 1-3 Half Adder Multisim

Full-Adders:

- A full-adder does binary addition on three inputs: A, B, and C_{in} .
- Full adders usually work in a cascade fashion where they are used to add binary numbers with an increasing number of bits.
- The two outputs are sum (S) and carry (C_{out}).
- You will notice that full adders can also use logic gates or a component.

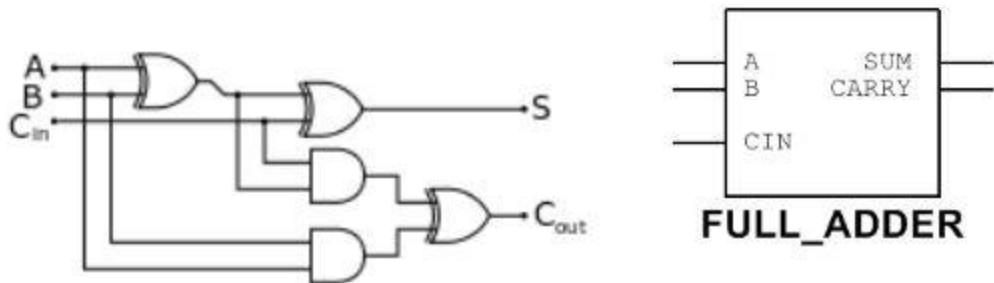


Figure 1-4 Full Adder from Gates

Figure 1-5 Full Adder Multisim

1-1 Why is it hard to use truth tables as the sequence of binary digits increases?

1-2 What are the similarities and differences between half adders and full adders?

1-3 When full adders are connected to each other such that C_{out} of one leads to the C_{in} of the other, this is called rippling. What do you think this means?

1.2 Simulate: Building a Half-Adder Circuit

Half-Adder Circuits

Half-adder circuits can be built using a combination of logic gates.

- Launch Multisim.
- Open a new circuit.
 - Select **File>>New**.
 - In the menu that appears, select **Blank** and click **Create**.
- Connect the following circuit:
 - Place an **XOR** gate and an **AND** gate from the **Misc Digital** group.
 - Place two **INTERACTIVE_DIGITAL_CONSTANTS** from the **Sources** group.
 - Place two **PROBE_DIG_REDs** from the **Indicators** group.
 - Wire them as shown:

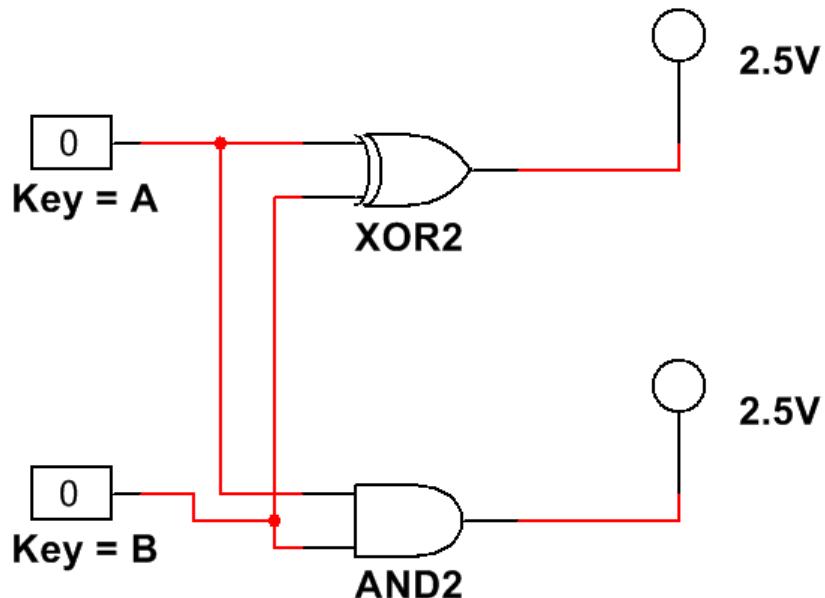


Figure 1-6 A Half Adder Circuit

- Click the **Run** button to begin simulating the circuit.



Figure 1-7 Run Button

- Using the **A** and **B** keys, vary the inputs into the circuit.

1-4 Fill out the truth table below.

A	B	XOR (SUM)	AND (CARRY)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$A'B + AB'$ AB

- Stop the simulation by clicking the **Stop** button.



Figure 1-8 Stop Button

Can you see how the XOR and AND gates represent the sum and carry of the numbers A and B added together?

- $0 + 0 = 0$
- $1 + 0$ or $0 + 1 = 1$, with no carry.
- $1 + 1 = 2$, but 2 is not a binary number. In binary, 2 is represented as 10. The 1 is the carry and the 0 is the sum.

1.3 Simulate: Building a Full-Adder Circuit

Full-Adder Circuits

Open a new PLD circuit.

Connect the following circuit:

- Place three **XOR** gates and two **AND** gates from the **Misc Digital** group.
- Place the input connectors **SW0**, **SW1**, and **SW2**.
- Place the output connectors **LED0** and **LED1**.
- Wire them as shown:

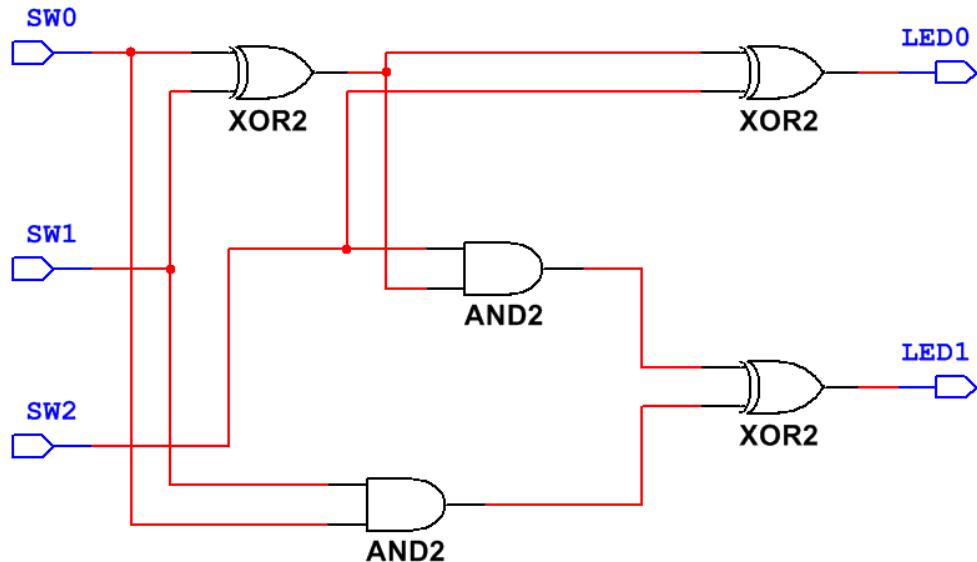


Figure 1-9 A Full Adder Circuit

- Export the circuit to the Digital Electronics Board. Using the switches on the board, vary the inputs into the circuit.

1-5 Fill out the truth table below.

SW0	SW1	SW2	SUM	C _{out}
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

$$\text{sum} = A'B'C + A'BC' + AB'C' + ABC$$

$$C_{\text{out}} = AC + AB + BC$$

Again, can you see how the truth table represents the sum and carry of the numbers A, B, and C_{IN} added together?

- 0 + 0 + 0 = 0
- 0 + 0 + 1 = 1, with no carry.
- 0 + 1 + 1 = 2, but 2 is not a binary number. In binary, 2 is represented as 10. The 1 is the carry and the 0 is the sum.
- 1 + 1 + 1 = 3. In binary, 3 is represented as 11. The first 1 is the carry and the second 1 is the sum.

1.4 Exercise: Building a Circuit Using Built-In Full Adders

Built-In Full Adders

Open a new circuit.

- Place a *full adder* on the circuit by clicking the **Place Misc Digital** button along the top bar.
- In the window that appears, select **TIL** from the **Family** selection box and scroll down to find **FULL_ADDER** in the **Component** box.
- Click **OK** and place the component on the left of the circuit board.
- The window will temporarily go away and then come back. When it comes back, it will already be set to place a full adder. Click **OK** and place the next adder to the right and bottom of the first (keep some space between the two).
- Continue this procedure and add two more adders to the circuit in the same pattern.
- Wire the **Carry** of the first adder from the left into the **C_{IN}** of the adjacent adder. Continue this procedure until all four adders are connected.

Note: The circuit should now look like this:

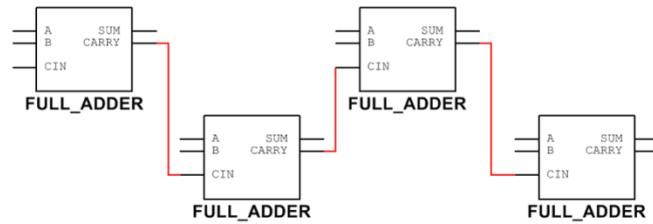


Figure 1-10 Building a Circuit Using Full Adders Step 1

We will now add the sources:

- Select the **Misc Digital** group.
- In the **Sources** group, select **DIGITAL_SOURCES** and **INTERACTIVE_DIGITAL_CONSTANT**.
- Select **OK** and place it on the circuit.
- When all sources have been placed, wire them as shown:

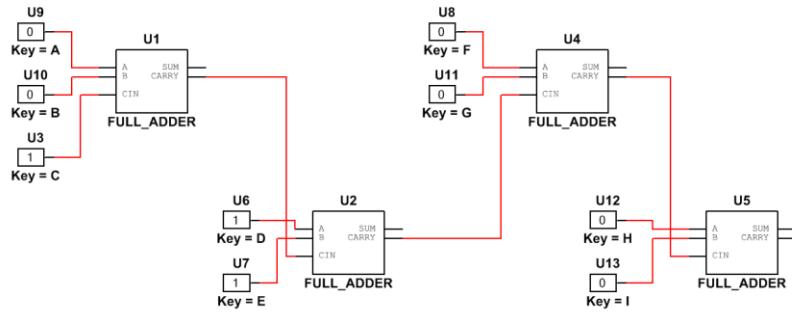


Figure 1-11 Building a Circuit Using Full Adders Step 2

Rippling Effects of Full Adders

Complete the circuit by adding one probe for each adder:

- Select the **Misc Digital** group.
- In the **Indicators** group, select **PROBE** and **PROBE_DIG_RED**.
- Select **OK** and place it on the circuit.
- When all probes have been placed, wire them as shown.
- Change the keys which control the different inputs, so that you may change their values independently.

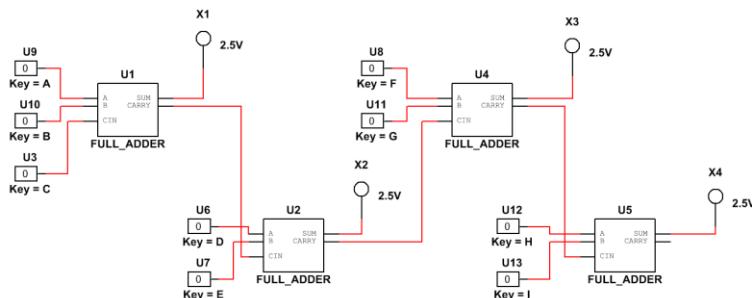


Figure 1-12 Building a Circuit Using Full Adders Step 3

- Click the **Run** button to begin simulating the circuit.



Figure 1-13 Run Button

1-6 Vary the inputs of the first adder to complete the following truth table.

A	B	C	SUM 1	SUM 2	SUM 3	SUM 4
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	1	1	0	1	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	0	0

1-7 Does this truth table match the one you tested in question 1-5?

yess

Note: Vary the inputs of adders 2-4 and record the combination that makes all the probes light up:



- When you are done, stop the simulation by clicking the **Stop** button.



Figure 1-14 Stop Button

1.5 Conclusion

1-8 Under what conditions would you use a half adder? A full adder?

1-9 What is the use of the Full Adder component within Multisim? In other words, why wouldn't you just place XOR and AND gates for each adder?

1-10 When converting binary numbers to BCD:

- A. Numbers are given specific values depending on their position in the sequence
- B. The value of the number on the left increases by twice the value of the number on the right
- C. Single digit numbers are already in BCD format
- D. All of the above

1-11 Half-adders do binary addition on two inputs using two logic gates. What is the correct combination of these logic gates?

- A. XOR and OR
- B. XOR and AND
- C. XNOR and AND
- D. NOT and OR

1-12 The outputs of a half adder are:

- A. Carry and C_{in}
- B. Sum and Carry
- C. Sum and C_{in}
- D. Carry and C_{out}

1-13 Which is a feature of full-adders?

- A. They have two inputs
- B. They have three outputs
- C. They work in a cascading fashion
- D. All of the above

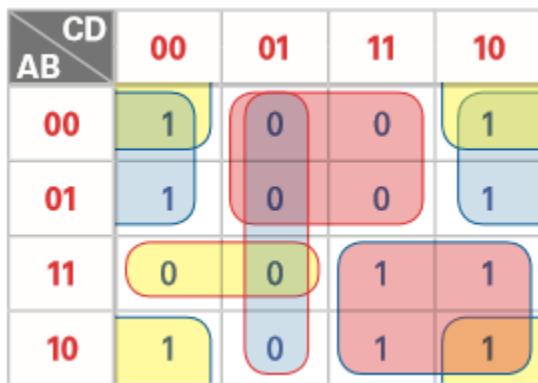
1-14 When connecting full-adders together in Multisim, the Carry of one is connect to _____ input of the next.

- A. A
- B. B
- C. C_{in}
- D. None of the above

Lab Manual:

Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



$$\text{SOP (1's): } AC + A'D' + B'D'$$

Kmap Colour

$$\text{POS (0's): } (A+D') \quad (C+D') \quad (A'+B'+C)$$

Kmap Colour

Lab 5: Karnaugh Maps

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 5: Karnaugh Maps

The *Karnaugh map (K-map)* is a tool and procedure used for minimizing Boolean functions. It is a graphical method that can be used for the manual design of simple logic functions having a small number of variables. K-mapping usually requires fewer steps than algebraic simplification and it always produces a minimum expression.

The Karnaugh map of a function is actually its truth table written as a grid. The rows and the columns of the map correspond to the possible values of the inputs and each cell represents the outputs of the function for the correlated inputs.

The simplified expressions are always in one of the two standard forms:

- Sum-of-Products
- Product-of-Sums

The cells are formed in a square or rectangle fashion and arranged such that neighboring cells have a single variable difference, otherwise known as *Gray code ordering*. For simplicity, the input values are placed as column and row labels. Each cell corresponds to a row in the truth table.

A	B	C	O
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

AB	C	0	1
00	0	1	
01	0	0	
11	1	0	
10	1	1	

Figure 1-1 Truth Table

Learning Objectives

In this lab, students will:

1. Simplify a Boolean expression using Karnaugh maps
2. Use a circuit with inputs to derive:
 - o The output experimentally and using Boolean algebra
 - o The Karnaugh map
 - o Simplified Combinational Logic Circuit

Required Tools and Technology

Platform: NI ELVIS III

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Software: NI LabVIEW FPGA Vivado 2014.4

- ✓ Install:
<http://www.ni.com/download/labview-fpga-module-2015-sp1/5920/en/>

Note: Digilent Driver (The installer above automatically downloads the installer below onto your computer)

- ✓ Navigate to:
C:\NIFPGA\programs\Vivado2014_4\data\xicom\cable_drivers\nt64\digilent
- ✓ Install: install_digilent.exe

Expected Deliverables

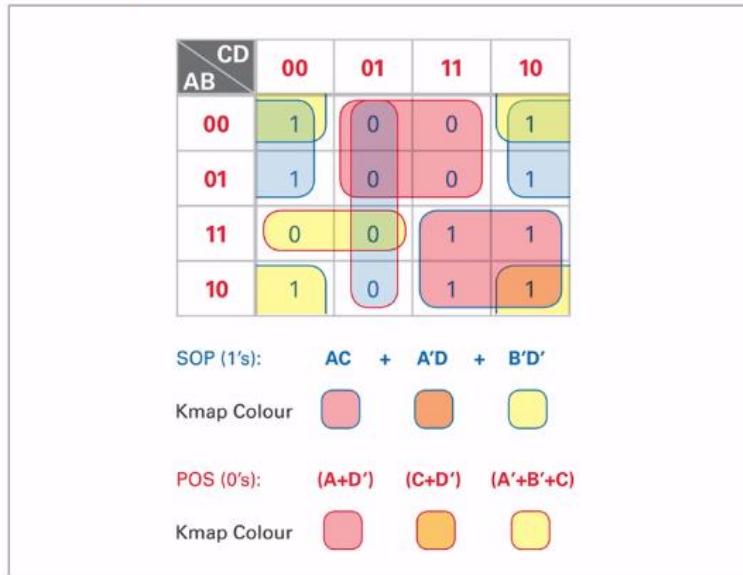
In this lab you will collect the following deliverables:

- Boolean expressions
- Analysis of gates for Combinational Logic Circuits
- Truth tables
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

K-Map



Simplify Boolean functions

- Simplify circuits
- Minimize material
- Reduce cost

Figure 1-2 Video. View the video here: <https://youtu.be/hVifB2hvJic>



Video Summary

- Karnaugh maps (K-maps) are used to simplify Boolean functions
- Simplified Boolean functions mean the number of logic gates needed is minimized
- K-maps are built by taking a truth table and converting it into a grid
- K-maps are useful if the logic functions have a small number of variables

After transferring the truth table to a Karnaugh map, cells with common output values, either all 0s or all 1s, are grouped into the largest possible rectangles. Cells are grouped in functions of 2^n either horizontally or vertically. Cells can be used more than once only if this generates the *least* number of groups. Also, all cells sharing the chosen common output must be contained within the grouping.

The groups generated can be converted to a Boolean expression. Each group represents a minimal minterm (1s) or maxterm (0s). The term is minimized by eliminating variables whose non-inverted and inverted forms appear within the same cell group. The terms left are then converted to a Boolean expression and used to derive an SOP or POS expression, which may then be converted to a combinational logic circuit.

When choosing the POS form, the variable combination is supposed to equal 0, as is the case when deriving a POS from a truth table. Therefore, the variable's inverse (1) is used when determining a maxterm.

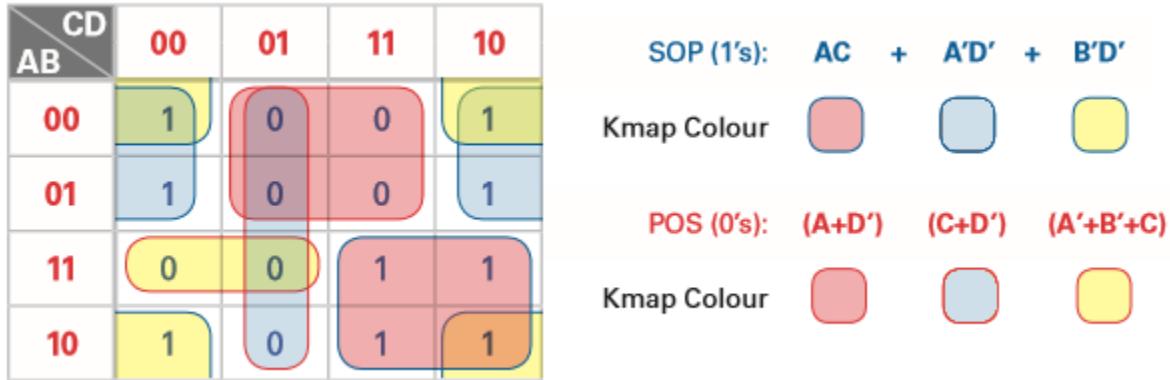


Figure 1-3 Karnaugh Map

Seven Segment Display

Karnaugh maps are useful for minimizing the number of logic gates needed in a circuit. In a practical sense, this reduction also results in a decrease in cost for a manufacturer since fewer components are needed to create an equivalent circuit.

A common way we work with Karnaugh maps is the *Seven Segment Display* (SSD). An SSD is an electronic device used for displaying numerical values. The device typically consists of seven segments arranged in a figure 8. Any digit, as well as some alphabet letters, can be displayed when the correct segments are activated. An example of an SSD as well as possible outputs can be seen in the image shown.

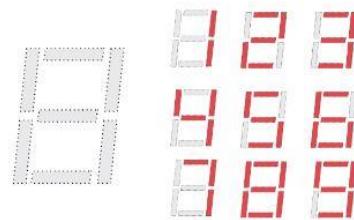


Figure 1-4 Seven segment display

Truth Tables

The truth table for an SSD consists of four inputs and seven outputs:

A	B	C	D	a	b	c	d	e	f	g	Numeric Output
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9
1	0	1	0	X	X	X	X	X	X	X	10
1	0	1	1	X	X	X	X	X	X	X	11
1	1	0	0	X	X	X	X	X	X	X	12
1	1	0	1	X	X	X	X	X	X	X	13
1	1	1	0	X	X	X	X	X	X	X	14
1	1	1	1	X	X	X	X	X	X	X	15

Figure 1-5 SSD Truth Table

Looking at this table, we can make several observations:

- Inputs are noted by capital letters (A-D).
- Outputs are denoted by lower case letters (a-g).
- The Numeric Outputs correspond to the numbers visible on the display.
- For numbers 10-15, “X” values are visible in the table. These are known as *don’t care conditions*.

The numerical outputs of 0-9 are necessary in an SSD, but outputs 10-15 are said to be illegal. In a Karnaugh map, don’t care conditions can be treated either as 0 or a 1, depending on which one produces a larger block. See below for an example.

$$a = B'C'D' + B'CD' + B'CD + BC'D + BCD' + BCD$$

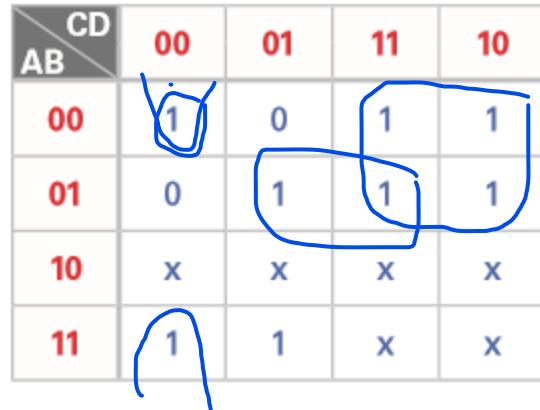


Figure 1-6 Karnaugh Map

$$\text{out} = ABC' + A'BD + A'C + C'D''$$

Note: Similar Karnaugh maps can be created for all other segments.

1-1 Create Boolean expressions (SOP and POS) from the Karnaugh map.

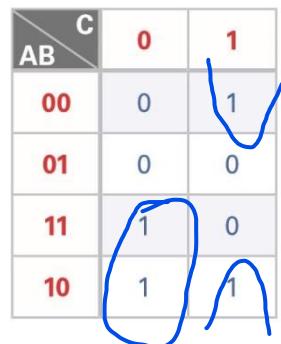


Figure 1-7 Karnaugh Map

$$AC' + B'C$$

1-2 Identify how many and which gates are needed to create the simplified Combinational Logic Circuit for the SOP expression.

1-3 Identify how many and which gates are needed to create the simplified Combinational Logic Circuit for the POS expression.

1.2 Exercise: Creating a Karnaugh Map from a Combinational Logic Circuit

PLD Design

Create the following circuit as a PLD design in Multisim:

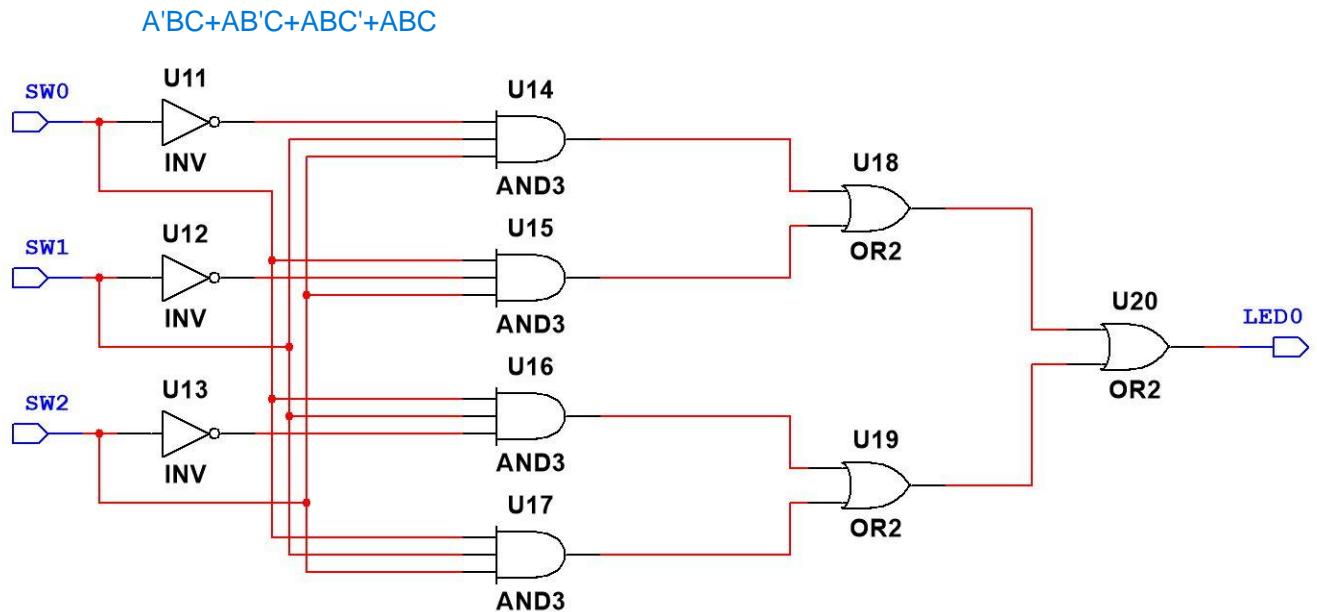


Figure 1-8 Circuit

- Export the circuit to the Digital Electronics Board.

1-4 Vary the inputs as per the truth tables and fill in the output.

SW0	SW1	SW2	Outputs
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\text{output} = A'BC + AB'C + ABC' + ABC$$

- Simplify the Output using Karnaugh maps. Take a screen shot, take a picture, or draw a sketch of your Karnaugh map and include it with your completed lab.
- Set up the simplified expression on Multisim. Take a screenshot, take a picture, or draw a sketch of the expression and include it with your completed lab.

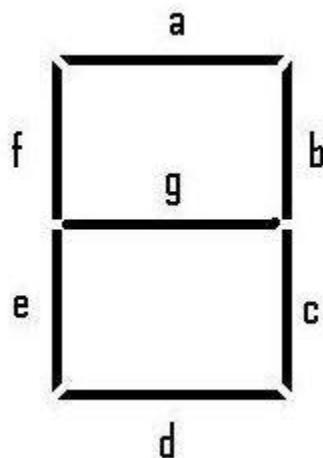
1-5 What is the resulting Output expression using Boolean algebra?

$$\text{output} = A'BC + AB'C + ABC' + ABC$$

1.3 Implement: Using Karnaugh Maps in Seven Segment Displays

Using Karnaugh Maps in Seven Segment Displays

- Using the truth table for a Seven Segment Display in the introduction, create Karnaugh maps for segments **a**, **b**, and **c**.
- Take a screenshot, take a picture, or draw a sketch of the maps and include it with your completed lab.



$$b = B' + C'D' + CD$$

Figure 1-9 Image for Karnaugh maps

$$a = B'C'D' + B'CD' + B'CD + BC'D + BCD' + BCD$$

- Create the simplified circuit for the 'a' segment in Multisim. Take a picture or screenshot and include it with your completed lab.
- Run the simulation in Multisim.

1-6 Does the behavior of the simplified circuit match the expected result? If not, make sure your Karnaugh map is simplified correctly, and the circuit matches the K-map.

- **Stop** the simulation when you are done.

1.4 Exercise: Simplified Circuits

Simplified Circuits

Instructions:

- Create the simplified circuit for the ‘b’ segment and repeat your test.
- Finally, create the simplified circuit for the ‘c’ segment and confirm its behavior.
- Take pictures or screenshots of these two circuits and include them with your completed lab.

1-7 How would you connect the 3 circuits simultaneously to produce the number 7?

- Create a circuit which will output 3 values to control the a, b, and c segments.
- Take a picture or screenshot and include this with your completed lab.

1-8 Would it be possible to simplify this circuit? If so, how?

1.5 Conclusion

1-9 Did you find it easier to create a combinational logic circuit from a truth table or a Karnaugh map? Explain.

1-10 Is there another device that can be used to create a seven-segment display? (Hint: Consider a BCD to Binary Decoder).

1-11 What are Karnaugh maps (K-maps) used for?

- A. Graphically minimizing Boolean functions
- B. Taking simple expressions of Boolean functions and expanding them
- C. Grouping opposite output values together (0's and 1's)
- D. Determining product-of-sums only

1-12 How many cells can be grouped together in the simplification of Karnaugh maps?

- A. No more than 4
- B. 2
- C. 2^n
- D. As many as possible

1-13 Creating Boolean expressions from Karnaugh maps typically leads to _____ in the number of logic gates needed in a circuit.

- A. An increase
- B. A decrease
- C. No change
- D. None of the above

1-14 How are 'don't' care conditions' treated in Karnaugh maps?

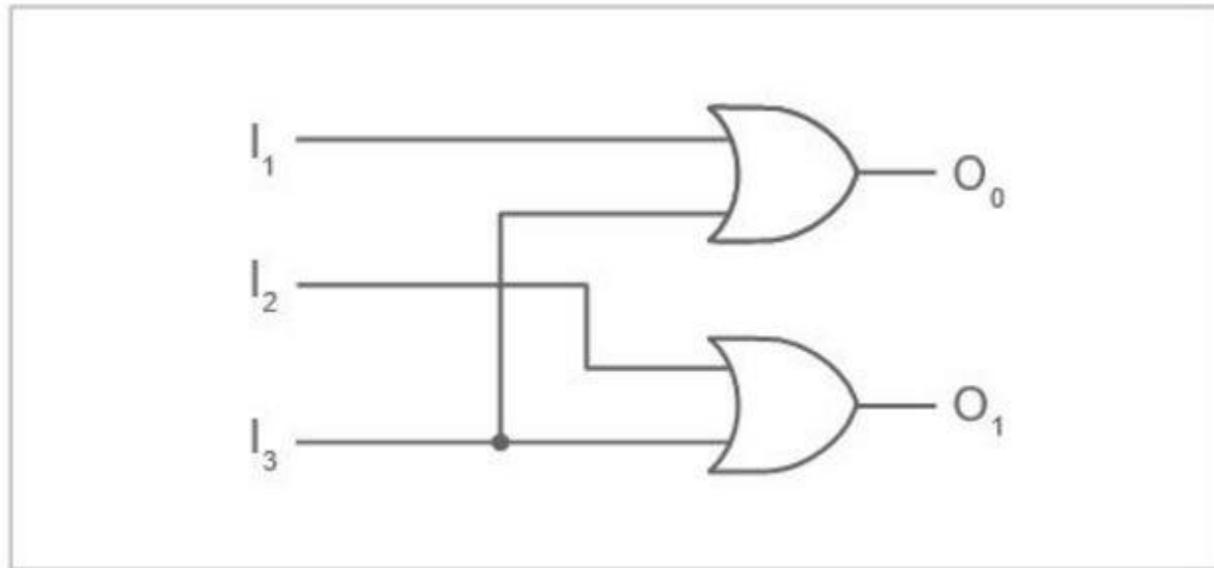
- A. They are ignored
- B. They are combined with the value on their right
- C. They can be treated as either a 0 or 1 depending on the situation
- D. They are always given a value of 0

1-15 A Seven Segment Display (SSD):

- A. Consists of seven segments arranged in a figure 8
- B. Can display any number between 0 and 9 digitally
- C. Can be simulated with logic gates or a clip
- D. All of the above

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 6: Encoders and Decoders

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 6: Encoders and Decoders

In Lab 4, we learned that gates arranged to perform a specific function, such as binary addition, can be represented with a chip. Other applications of this concept include encoders and decoders. Encoders are logic circuits responsible for reducing the size of an input. Decoders perform the inverse operation, increasing the size of an input. In a previous lab, we learned what a seven-segment display is and in this lab we will explore how decoders apply to this electronic device.

Learning Objectives

In this lab, students will:

1. Explain how decoders work, specifically in an SSD
2. Create a circuit with a BCD to Seven Segment Display Decoder and verify its truth table
3. Create a circuit with a pro

Required Tools and Technology

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Expected Deliverables

In this lab you will collect the following deliverables:

- Diagram of a BCD 7 Segment Display Decoder
- Truth tables
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for requirements and templates.

1.1 Theory and Background

Decoders

n bits			2 ⁿ bits			
En	I ₁	I ₂	O ₀	O ₁	O ₂	O ₃
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0
0	X	X	0	0	0	0

Translate binary code

One hot encoded:

- For any combination of inputs, only one output is 1

The value of the enable signal dictates if the decoder will work

Used for memory access

Figure 1-1 Video Screenshot. View the video here: https://youtu.be/RH2SeKV_DKg



Video Summary

- Decoders are devices that translate a binary code of n-bits of information into 2^n bits of information
- For any combination of signals there is only one output that can have a value of 1
- Encoders perform the opposite function of a decoder
- Encoders encode information from 2^n input lines into an n-bit output line

Decoders

The process of translating ambiguous information into something understood by a device receiving the data is called *decoding*.

Therefore, the resulting device is known as a *decoder*.

- Decoders take binary codes of n bits and generate 2^n outputs.
- The outputs of a binary decoder are said to be *one-hot encoded* because for any combination of the input signals there is only one output having the value 1.
- Decoders can include an *enable signal* for controlling the circuit operation.
- This enable signal can be active-low (meaning that the circuit will operate only when enable is 0) or active-high (the decoder is enabled when enable is 1).
- Decoders with enable inputs can be used for constructing larger decoders.
- One of the most important applications of decoders is memory access, where they are used for decoding the address of the rows in the memory blocks.

Let's take the example of a 2 to 4 decoder enabled when the *En* signal is 1:

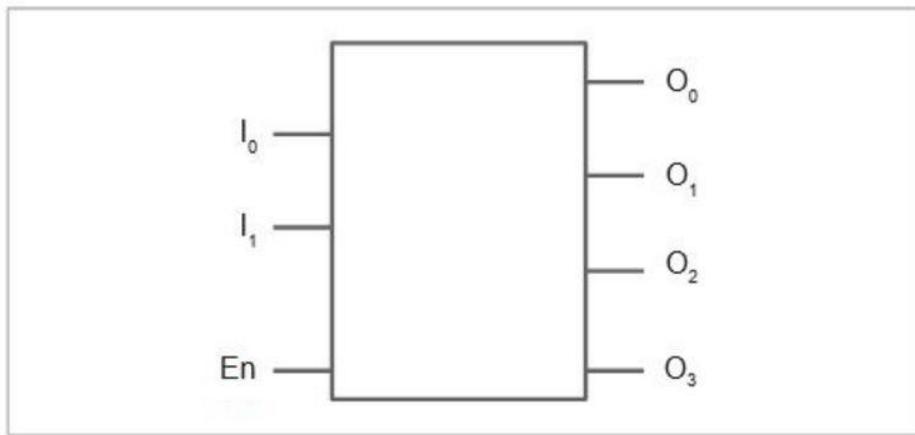


Figure 1-2 Decoder

From this, we can determine the following truth table and logic circuit:

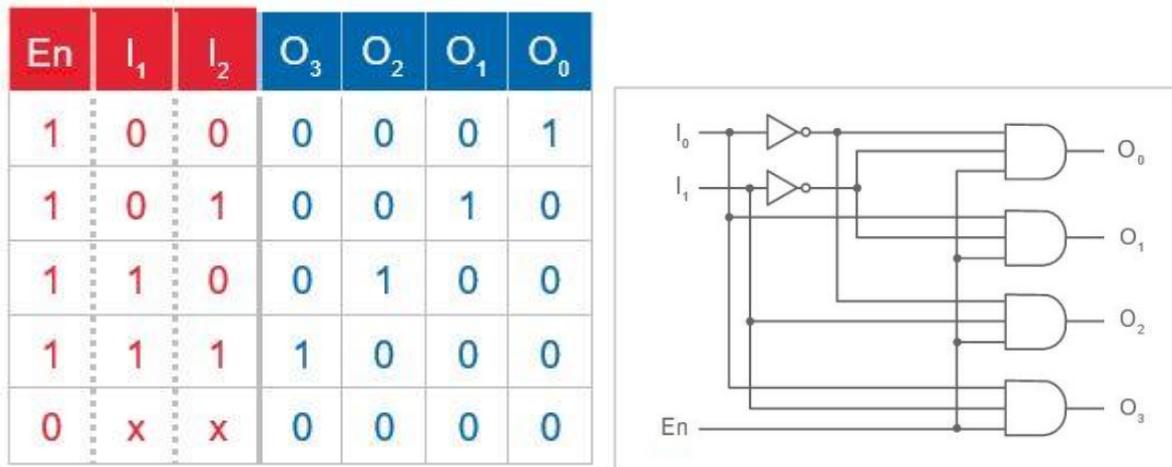


Figure 1-3 Truth table (left) and logic circuit (right)

Encoders

Encoders are logic circuits that perform the opposite function of a decoder. Binary encoders encode information from 2^n input lines, producing an n-bit code.

- At any given time, only one of the 2^n inputs can be 1.
- Encoding is used for reducing the number of bits needed to represent information. They are often used in application such as data transmission and data storing.
- The graphical symbol of the 4 to 2 binary encoder is presented below. The cases in which more than one input is 1 are not shown in the truth table because they are treated as don't care conditions.

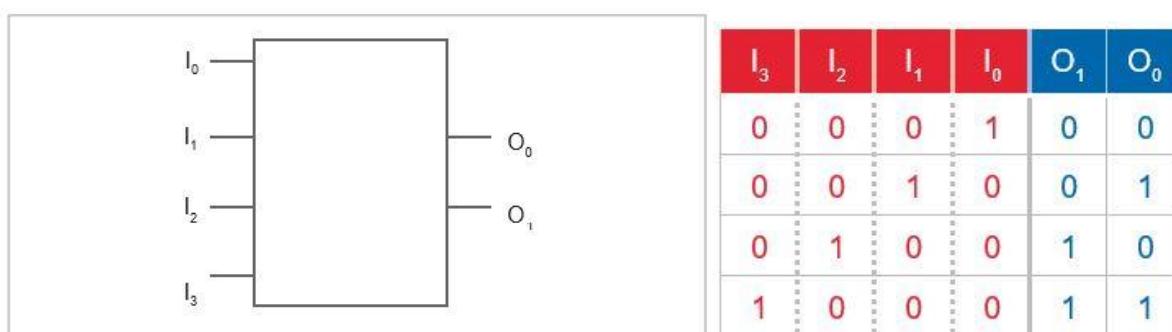


Figure 1-4 Encoder (left) and truth table (right)

- It can be seen in the truth table that the output O_1 is 1 when either I_3 or I_2 is 1 and that the output O_0 is 1 when either I_3 or I_1 is 1.
- It can also be seen that the input I_0 can be ignored
- The encoders presented so far are considered to have *one-hot encoded* inputs.

The corresponding logic circuit is presented below:

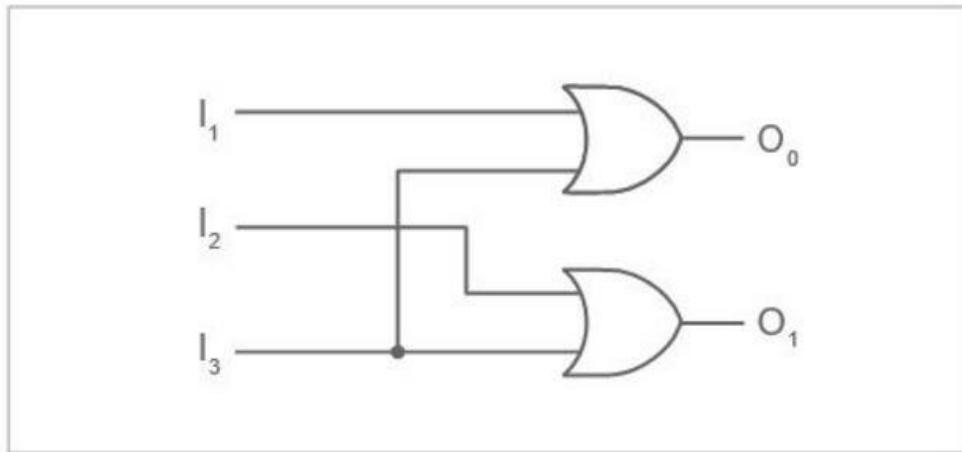


Figure 1-5 Logic circuit

Another commonly used type of encoder is a *priority encoder*:

- Priority encoders are able to prioritize inputs
- This is important because regular encoders can generate the wrong output when there is more than one input present at logic level 1.
- This type of encoder has an additional output, z , which indicates the case in which none of the inputs is 1.

The graphical symbol of the priority encoder is presented below. The truth table describes the behavior of a 4-to-2 priority encoder. It can be seen on the last line of the truth table that if the input I_3 is 1, the outputs are all 1 and the values on the other inputs of the decode do not matter and are denoted by 'x'.

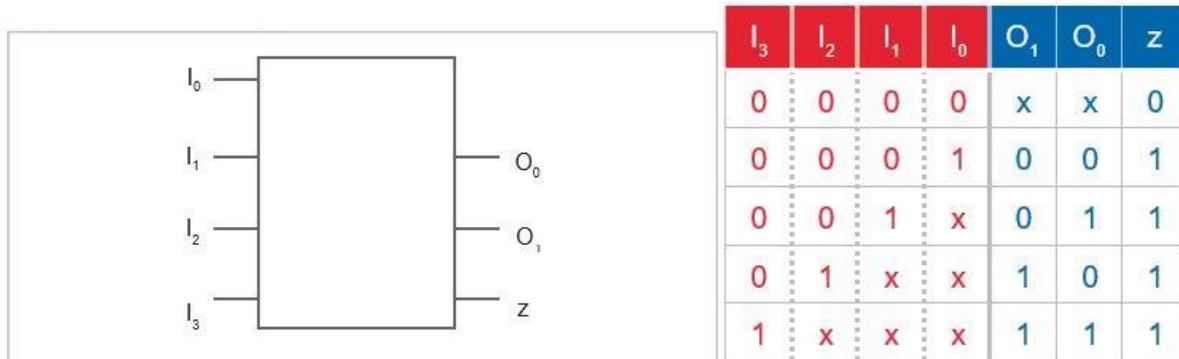


Figure 1-6 Priority encoder (left) and truth table (right)

1.2 Exercise: Seven Segment Display Diagram

Note: Seven Segment Displays (SSD) explored in the previous lab use a Binary Coded Decimal (BCD) to a 7-Segment Display Decoder.

1-1 Explain briefly how a decoder works.

- Draw a diagram of the BCD to 7-Segment Display Decoder and provide a picture or snapshot with your completed lab.

1-2 What is the difference between an encoder and a priority encoder?

1.3 Implement: Building a BDC to Seven Segment Display Decoder

BCD to Seven Segment Display Decoder

Build the following circuit:

- Click the **Misc Digital** button and from the **TTL** Group, select the **7447N Decoder**.
- Click the **Misc Digital** button and from the **Basic** Group, select the **RPACK Family** and then the **7Line_Isolated** resistor.
- Right click on the resistor and view **priorities**. Change the resistance to **220 Ω**
- Click the **Misc Digital** button from the **Indicators** Group, select **HEX_DISPLAY** and then **SEVEN_SEG_COM_A_GREEN**.
- Click the **Misc Digital** button from the **Sources** Group select **POWER_RESOURCES** and then **VCC**. Place one near the bottom of the Decoder and one near the top of the SSD.
- Place four **INTERACTIVE_DIGITAL_CONSTANTS**.
 - Change the keys for toggle to match the ones shown in the figure below.

Write them as shown:

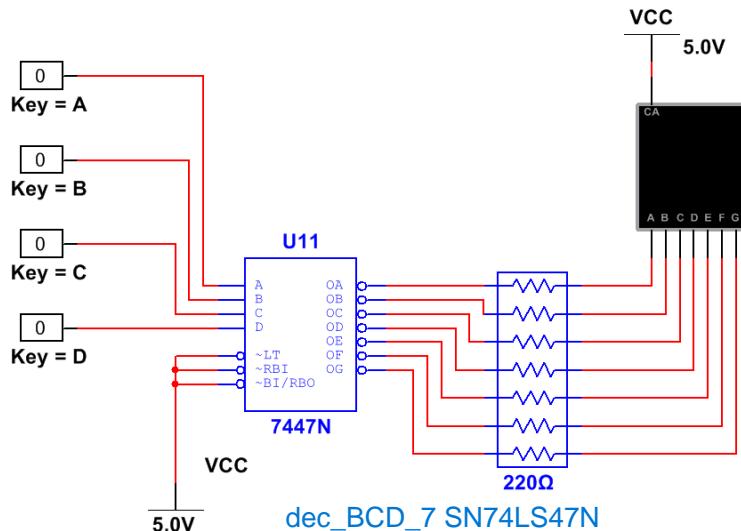


Figure 1-7 Circuit diagram

Testing a BCD to Seven Segment Display Decoder

- Run the Simulation

1-3 Vary the inputs to complete the following truth table of an SSD

A	B	C	D	a	b	c	d	e	f	g	Numeric Output
0	0	0	0								0
0	0	0	1								8
0	0	1	0								4
0	0	1	1								
0	1	0	0								2
0	1	0	1								
0	1	1	0								6
0	1	1	1								
1	0	0	0								1
1	0	0	1								9
1	0	1	0	X	X	X	X	X	X	X	none 5
1	0	1	1	X	X	X	X	X	X	X	none
1	1	0	0	X	X	X	X	X	X	X	none 3
1	1	0	1	X	X	X	X	X	X	X	none
1	1	1	0	X	X	X	X	X	X	X	none 7
1	1	1	1	X	X	X	X	X	X	X	none

1.4 Implement: Building a Circuit for a 10 to 4 Priority Encoder

Circuit

- Create the following circuit:

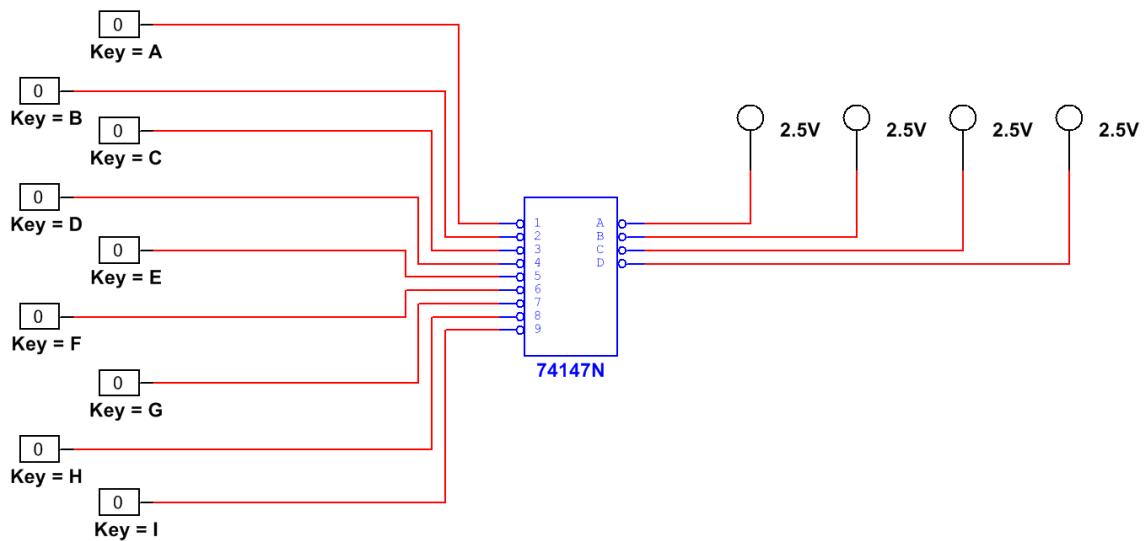


Figure 1-8 Circuit diagram

Testing 10 to 4 Priority Encoder

- Run the circuit.

1-4 Complete the following truth table for the 10 to 4 priority encoder.

Note: Knowing that only one input is supposed to be low at a time for this particular encoder, there are many rows of the truth table that you can skip.

A	B	C	D	E	F	G	H	I	O1	O2	O3	O4
0	0	0	0	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	0	0	1	1	0
0	1	0	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	1	0	0	0	0	0	0	1	1	0
0	0	0	0	1	0	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	1	1	0
0	0	0	0	0	0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0	1	1	1	1	0

1-5 What happens when two inputs are low?

1-6 Can you determine how the encoder defines priorities?

1-7 What gates would need to be created to reproduce this 10 to 4 encoder, given the truth table?

- Stop the circuit when you're done.

1.5 Conclusion

1-8 Why do encoders and decoders have different ratios of inputs and outputs?

1-9 Why is it that the encoder has an input line that can be ignored when defining behavior with gates?

1-10 How do Enable signals control the way a decoder works?

- A. It decides which of the inputs enters the decoders first
- B. It converts n bits to 2^n outputs
- C. Determines whether the circuit will operate when it is set to either 0 or 1
- D. All of the above

1-11 Which logic gates are needed to create a 2 to 4 decoder?

- A. 2 NOT gates and 4 AND gates
- B. 2 NOT gates and 4 OR gates
- C. 2 AND gates and 4 NOT gates
- D. 2 AND gates and 4 OR gates

1-12 In binary encoders, at any given time:

- A. Only one of the n outputs can be 1
- B. Only one of the $2n$ inputs can be 1
- C. There must be an enable input
- D. Can store a maximum of 2 bits of information

1-13 How is a priority encoder different from a one-hot coded one?

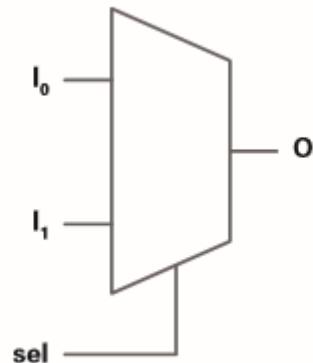
- A. It prioritizes inputs that are 0
- B. It prioritizes the I1 input
- C. It prioritizes which input is more important if there is more than one at logic level 1
- D. It prioritizes which input is more important if there is more than one at logic level 0

1-14 Encoders use which type of logic gate to accomplish their behavior?

- A. A NAND gates
- B. NOR gates
- C. XOR gates
- D. OR gates

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 7: Multiplexers and Demultiplexers

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 7: Multiplexers and Demultiplexers

Multiplexers are combinational logic circuits for which there are multiple potential inputs but there is always only one output. Demultiplexers are the opposite in that there is always one input but there are multiple potential outputs. Both multiplexers and demultiplexers have a bit (or multiple bits) called selector bit(s) which is responsible for determining which input or output is chosen. Like encoders and decoders, multiplexers and demultiplexers can be broken down into circuit components but are typically represented by chips for visual simplification. In this lab, we will analyze multiplexers and demultiplexers in both their circuit and chip forms.

Learning Objectives

In this lab, students will:

1. Reflect on the similarities and differences between encoders and multiplexers
2. Examine the function of a basic 2-to-1 Multiplexer using logic gates
3. Observe the behavior of clock multiplexing using an oscilloscope

Required Tools and Technology

Platform: NI ELVIS III

Instruments/Parts used in this lab:

- Oscilloscope
- Wire

Note: The NI ELVIS III Cables and Accessories Kit (purchased separately) is required for using the instruments

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M
- ✓ Install Soft Front Panel support:
<http://www.ni.com/documentation/en/ni-elvis-iii/latest/getting-started/installing-the-soft-front-panel/>

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Software: NI LabVIEW FPGA Vivado 2014.4

- ✓ Install:
<http://www.ni.com/download/labview-fpga-module-2015-sp1/5920/en/>

Note: Digilent Driver (The installer above automatically downloads the installer below onto your computer)

- ✓ Navigate to:
C:\NIFPGA\programs\Vivado2014_4\data\xicom\cable_driver

s\nt64\digilent

- ✓ Install: install_digilent.exe

Expected Deliverables

In this lab you will collect the following deliverables:

- Sum-of-Products Boolean functions for 2-to-1 Multiplexer
- Sum-of-Products Boolean functions for 1-to4 Demultiplexer
- Screenshot of Venn diagram
- Image of circuit
- Observations of demultiplexer behavior
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

Multiplexers

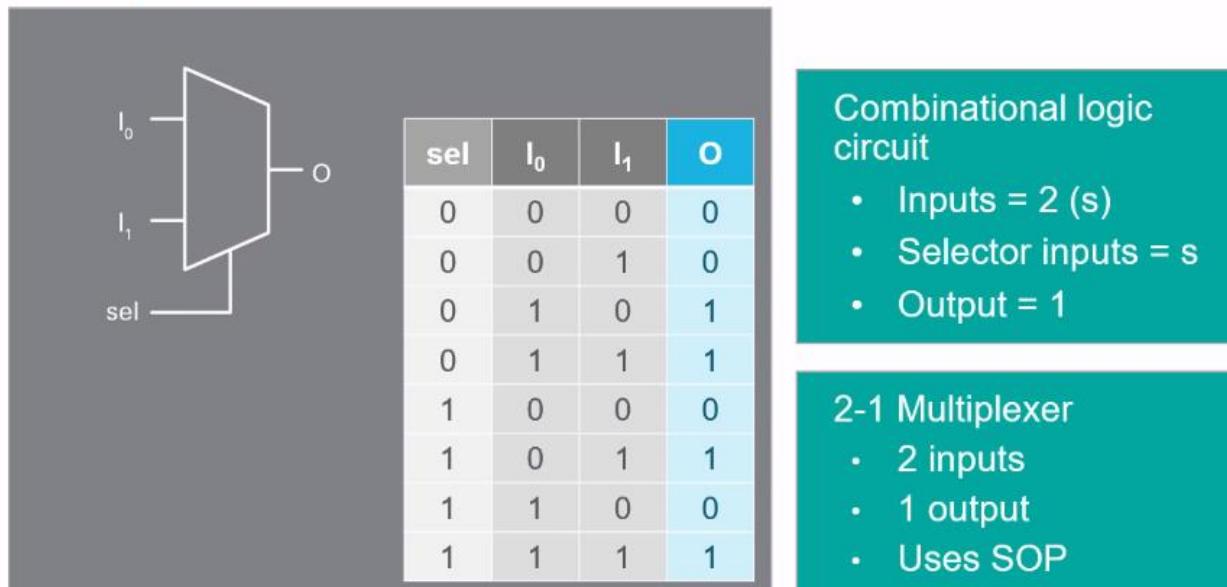


Figure 1-1 Video. View the video here: https://youtu.be/khmQ-LT_Cxg



Video Summary

- Multiplexers are combinational logic circuits
- Clock multiplexing is used for operating the same logic function at different clock rates from different sources
- Demultiplexers are combinational logic circuits that have the opposite function of a multiplexer

Multiplexers

The *multiplexer*, abbreviated *MUX*, is a combinational logic circuit which has multiple data inputs, one or more select inputs and one output.

- It passes the data on one of the inputs, depending on the selection signals, to the output
- With the help of this logic circuit, multiple signals can share the same data output
- Multiplexers have 2^s inputs and s selector lines, which determine which of the inputs to output.
- Multiplexers are one of the most widely used combinational circuits, their application areas include:
 - Data routing
 - Operation sequencing
 - Parallel-to-serial conversion
 - Waveform generation

The simplest circuit is the 2-to-1 multiplexer, with the graphical symbol presented in the leftmost figure. Its functionality is described by the joining truth table. The multiplexer below is only 1-bit wide since bit line is connected to a single output bit line.

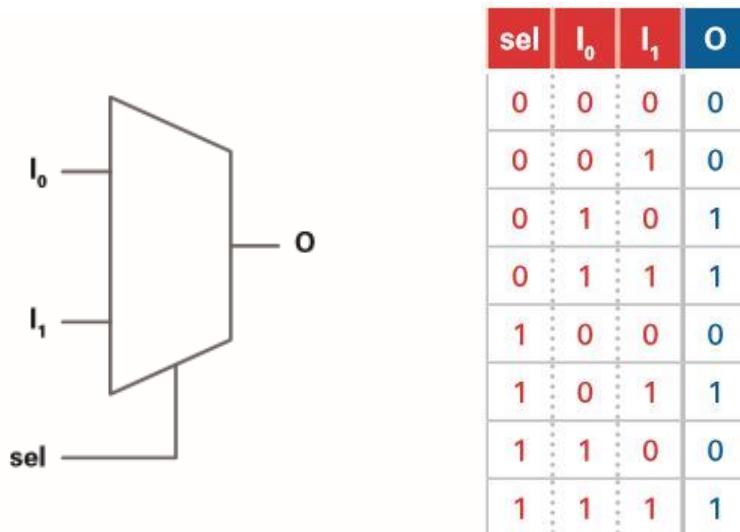


Figure 1-2 Image of 2-to-1 multiplexer (left) and truth table (right)

The truth table can be simplified to the following truth table for a better understanding of the circuit's operation:

sel	O
0	I_0
1	I_1

Figure 1-3 Simplified truth table

Using the sum-of-products Boolean function gives the following combinational logic circuit:

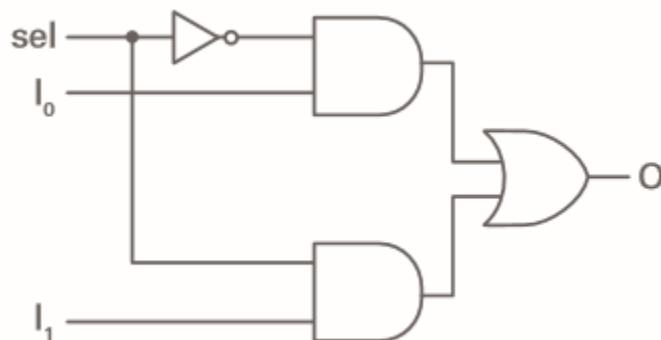


Figure 1-4 Combinational logic circuit

Clock multiplexing is a technique used for operating the same logic function at different clock rates, from different sources (inputs).

- The logic circuits are switched by the select signal often while the circuit is running
- This process of switching isn't very safe and can result in a glitch that occurs when one signal is going down as the other is going up.
- Clock safe switches can be implemented to eliminate glitches.

Demultiplexers

Demultiplexers (DEMUX) have the opposite function of a multiplexer

- It places the value of a single data input on several data outputs depending on a selection signal
- Usually demultiplexers have s select inputs and 2^s outputs
- Since demultiplexers take one input and connect it to many outputs, some of their uses are for communication (two-way communication usually includes both multiplexers and demultiplexers) and for serial to parallel converters
- The graphical symbol for a 1-to-4 demultiplexer is shown below (left) as well as the corresponding 1-to-4 DEMUX truth table (centre) and the CLC (right)

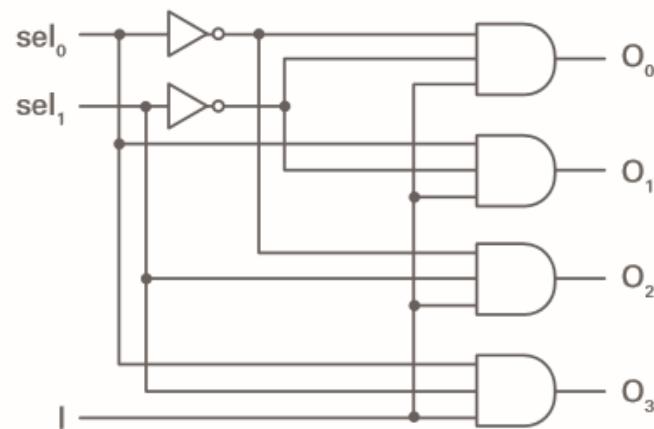
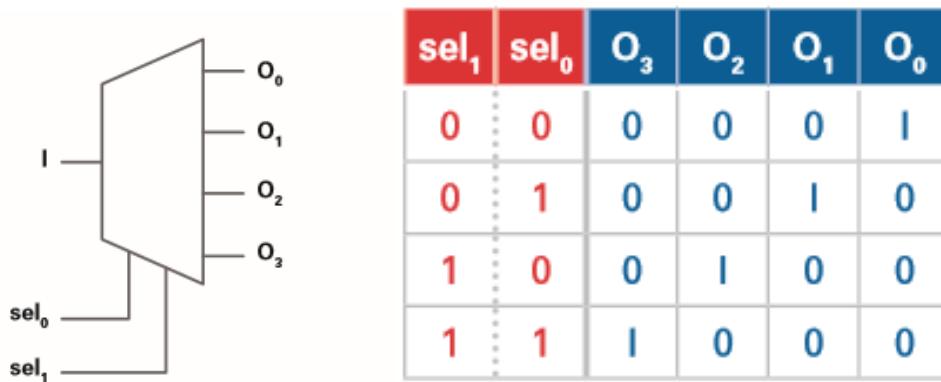


Figure 1-5 Demultiplexer (top left), truth table (top center) and CLC (bottom)



Check Your Understanding

Note: The following questions are meant to help you self-assess your understanding so far. You can view the answer key for all “Check your Understanding” questions at the end of the lab.

1-1 Write the sum-of-products Boolean functions for the 2-to-1 Multiplexer:

1-2 Write the sum-of-products Boolean functions for the 1-to-4 Demultiplexer:

1-3 What is the function of the Selector (Sel) in Multiplexers and Demultiplexers?

- Use a Venn diagram to show the similarities and differences between Encoders and Multiplexers. Add the file, picture, or a screenshot of the Venn diagram to your completed lab.

1.2 Implement: Multiplexers Using Logic Gates

Circuit 1

Build and connect the following circuit in a PLD design:

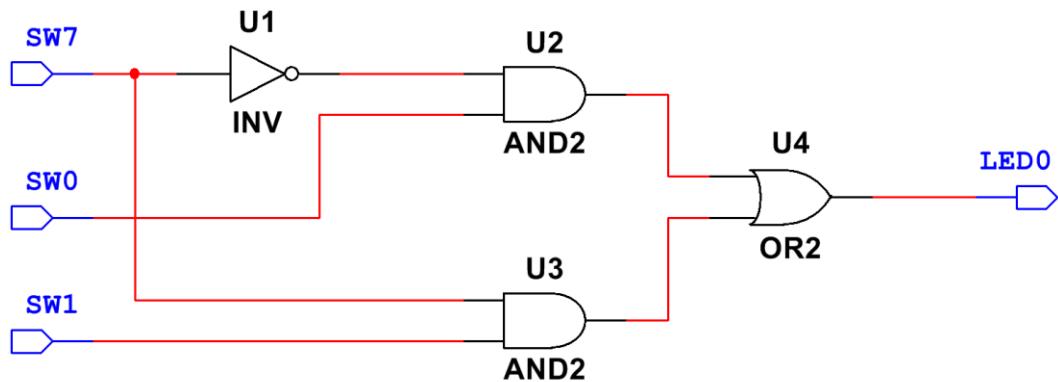


Figure 1-6 Circuit diagram

- Export the circuit to the Digital Electronics Board
- Set the selector (SW7) to 0.
- Toggle switches **SW0** and **SW1**. Notice that the output is determined by SW0 and the behavior of SW1 has no effect on the output.
- Set the **Selector** to 1.
- Toggle switches **SW0** and **SW1**. Notice that the output is determined by the value of SW1 and the value of SW0 has no effect on the output.
- Take a screenshot, sketch, or take a picture of your circuit and attach it to your completed lab.

4-to-1 MUX

Using the following truth table (right) to describe the behavior of a 4-to-1 MUX (left), design and implement the corresponding CLC.

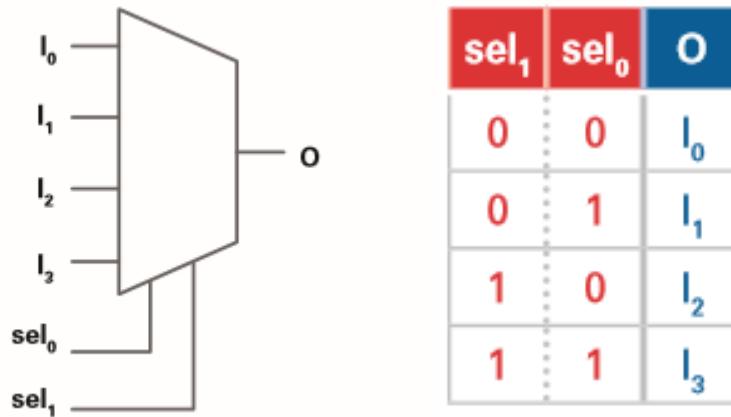


Figure 1-7 Image of 4-to-1 MUX (left) and truth table (right)

- Take a screenshot, sketch, or take a picture of the CLC you create and add it to your completed lab.

1.3 Implement: Clock Multiplexing

Circuit 2

Build and connect the following circuit in a PLD design:

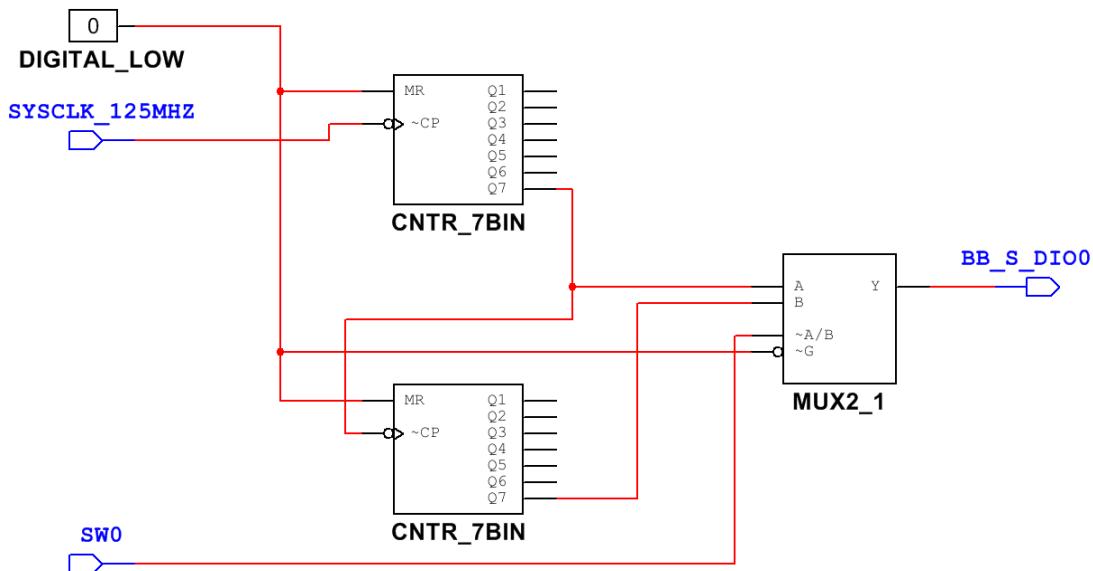


Figure 1-8 Circuit diagram

Notice how much quicker it is to implement the 2-to-1 MUX as a single block. Most common CLCs are available as chips, because they are the building blocks of digital electronics.

- Export the circuit to the Digital Electronics Board.
 - The 125 MHz system clock is processed by the two counters and two different rate clock signals, one having the frequency of 977 kHz, and the other having the frequency of 8 kHz, are generated.
- Connect **BB_S_DIO0** to an oscilloscope
- Set **SW0** to **0** and observe the output signal of the oscilloscope.

1-4 How would you describe the oscilloscope output signal?

- Set **SW0** to 1 and observe the output signal of the oscilloscope?

1-5 How would you describe the oscilloscope output signal?

- Attach an image of your final circuit. If possible, attach a picture of both oscilloscope output signals as well to your completed lab.

1.4 Implement: Demultiplexer

1-to-4 Demultiplexer

In either simulation or on the Digital Electronics Board, build and run the following 1-to-4 demultiplexer

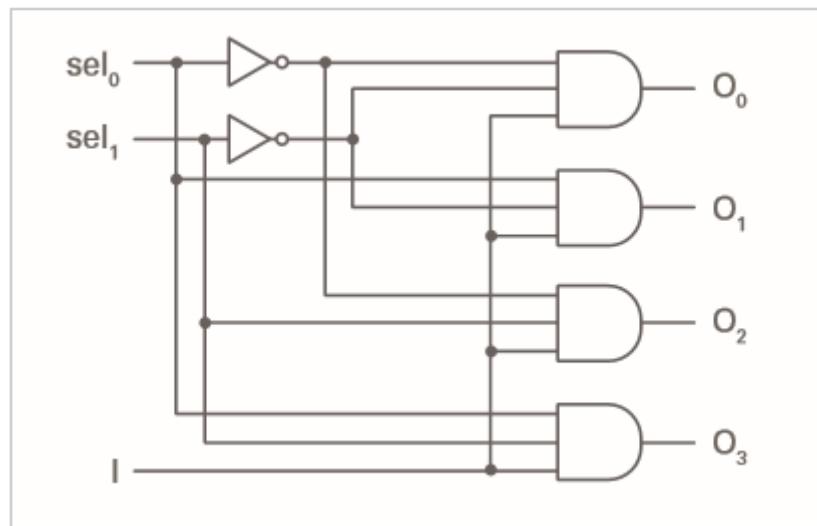


Figure 1-9 Image of 1-to-4 demultiplexer

- Given the truth table for a 1-to-4 demultiplexer below, convert the selectors into binary numbers.

sel_1	sel_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Figure 1-10 Truth table

- Notice that the binary numbers indicate which output will be On.
- Confirm that your circuit follows this behavior, and record your observations and include them with your completed lab.

1-6 How would you implement a 1-to-8 demultiplexer? Define the pattern for adding more outputs?

1.5 Conclusion

1-7 How did the patterns of the oscilloscope vary depending on the selector value?

1-8 What were some the advantages and disadvantages of using the Digital Electronics Board rather than simulating the above circuits?

1-9 In an everyday application, when would a multiplexer be useful?

1-10 In an everyday application, when would a demultiplexer be useful?

1-11 How many outputs does a multiplexer have?

- A. 1
- B. 2
- C. 3
- D. 2^n

1-12 Why can the truth table of a 2-to-1 multiplexer be simplified depending on whether the selector is set to 0 or 1?

- A. There is only one output
- B. Some of the outputs of the original truth table are don't care conditions
- C. The line that is selected to be inputted will be the only one affecting the output
- D. None of the above

1-13 A clock multiplexer:

- A. Performs a multiplexing action for a specified amount of time
- B. Is switched between inputs by the select signal at a specified rate
- C. Functions most effectively when it has 4 inputs
- D. Needs to be connected to a resistor to slow the current to a specified rate

1-14 The 1-to-4 demultiplexer has how many selectors?

- A. 4
- B. 3
- C. 2
- D. None of the above

1-15 What is the difference between the logic circuit of a 2-to-4 decoder and a 1-to-4 demultiplexer?

- A. They use a different combination of logic gates
- B. They have a different number of outputs
- C. They have different inputs
- D. All of the above

Answer Key – Check Your Understanding Questions Only



Check Your Understanding

1-1

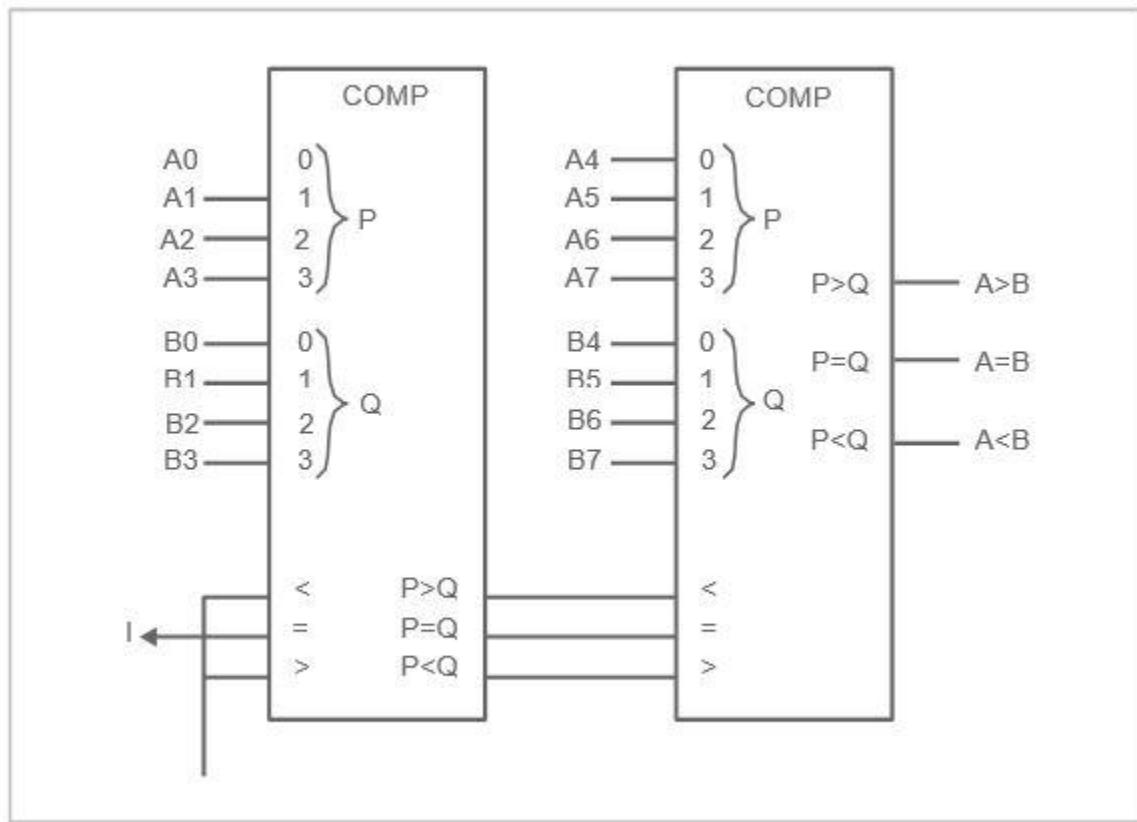
1-2

1-3

Lab Manual:

Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 8: Comparators

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 8: Multiplexers and Demultiplexers

In this lab we will learn how to implement the <, > and = operations in a combinational logic circuit. One comparator can compare two one-bit binary numbers. However, similarly to adders, numbers of more than one bit can be analyzed if multiple comparators are joined together.

In previous labs we have discussed the value of representing CLCs like adders and comparators with their chip equivalent especially with regards to visual simplification. In this lab, we will explore sub-circuits and how they relate to this concept.

Learning Objectives

In this lab, students will:

1. Gain exposure to the purpose of a sub circuit within a larger circuit in the Multisim PLD schematic.
2. Recognize that comparators can be constructed with logic gates or chips that can be combined if/when the number of bits increases.
3. Reflect on the similarities and differences between adders and comparators.

Required Tools and Technology

Platform: NI ELVIS III

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Software: NI LabVIEW FPGA Vivado 2014.4

- ✓ Install:
<http://www.ni.com/download/labview-fpga-module-2015-sp1/5920/en/>

Note: Digilent Driver (The installer above automatically downloads the installer below onto your computer)

- ✓ Navigate to:
C:\NIFPGA\programs\Vivado2014_4\data\xicom\cable_drivers\nt64\digilent
- ✓ Install: install_digilent.exe

Expected Deliverables

In this lab, you will collect the following deliverables:

- Screenshot of a combinational logic circuit for a 4-bit comparator
- PLD sub-circuit in the Workspace
- Observations about 4-bit parallel comparators
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

What are Magnitude Comparators?

Inputs		Outputs		
A	B	A>B	A=B	A<B
0	0	0	1	0
1	0	1	0	0
0	1	0	0	1
1	1	0	1	0

Compare word A and word B:

- A=B
- A>B
- A<B

The A=B column is equivalent to an XNOR gate

Figure 1-1 Video. View the video here: <https://youtu.be/BP5G3n9dmYA>



Video Summary

- The A = B column in the truth table is equivalent to an XNOR gate
- Large cascading comparators have 4-bit inputs

In its simplest terms, *comparators* are combinational logic circuits that are used to test whether word A is less than ($<$), equal to ($=$) or greater than ($>$) word B. Comparators that determine whether one value is less than, equal to or greater than another are called magnitude comparators. The truth table for a 1-bit digital *magnitude comparator* can be seen below:

Inputs		Outputs		
A	B	A>B	A=B	A<B
0	0	0	1	0
1	0	1	0	0
0	1	0	0	1
1	1	0	1	0

Figure 1-2 Truth table

Upon examining the table, it can be seen that the A=B output row corresponds to that of the XNOR gate. XNOR gates are used in the design of digital comparators. Below is the logic circuit for the truth table above:

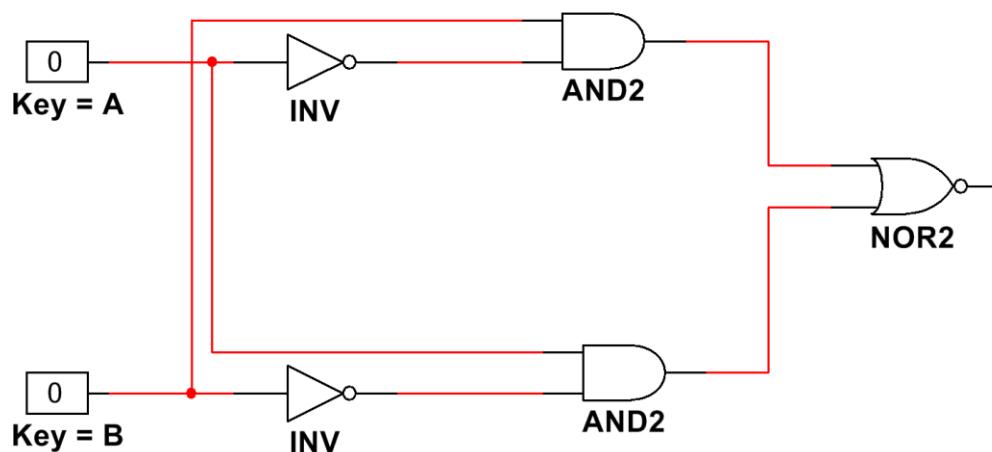


Figure 1-3 Logic circuit diagram

Similar to adders, larger comparators can be designed in a cascading fashion to accommodate n number of inputs. For example, let us consider two 4-bit magnitude comparators (shown below). We can see the 4-bit inputs are (A₀, A₁, A₂, A₃) and (B₀, B₁, B₂, B₃). P corresponds to the word generated by all of the A inputs. Q corresponds to the word generated by all of the B inputs. The magnitude comparator begins by comparing the *highest order bit (MSB)* first and then works its way down from the next highest bit until it reaches the *lowest-order bit (LSB)*. Once the lowest order bits have been compared, the comparator stops. Digital Comparators are often used in Central Processing Unit (CPU) of computers and microcontrollers.

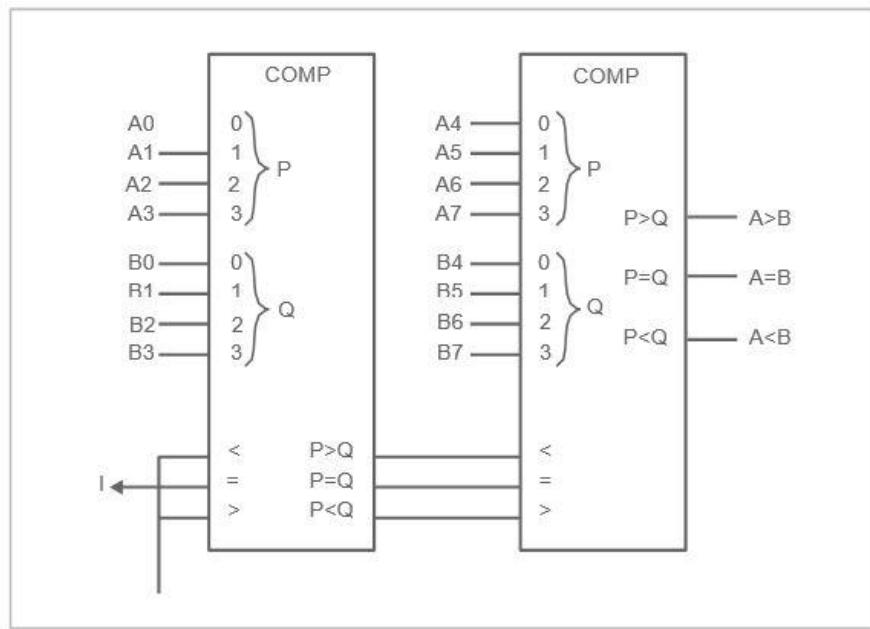


Figure 1-4 Comparator diagram

1-1 How do MSBs and LSBs determine the sequence of bit comparisons?

- Using the CLC from the introduction as a reference, draw a combinational logic circuit for a 4-bit comparator. Attach your sketch to your completed lab.

1.2 Implement: Building a 4-bit Parallel Comparator

Part 1

Open Multisim and select **File >> New >> Blank** to create new design.

- Select **Place » New PLD** sub circuit.
- Select the NI Digital Electronics Board from the drop-down menu and click **Next**.
- Name the sub circuit Comparator and confirm the appropriate PLD part number, then click **Next**.
- Select the following connectors: **SW0** to **SW7** which correspond to the slide switches in the physical board. Bits A0 to A3 are represented by SW0 to SW3, and bits B0 to B3 by SW4 to SW7.
- Select **LEDO** for the output of the circuit.
- Click **Finish** and place the PLD sub-circuit on the workspace.

Part 2

Select **Place >> Component** and place the following components in the workspace:

Component	Group	Family
VCC	Sources	POWER_SOURCES
DSWPK_8	Basic	SWITCH
PROBE_ORANGE	Indicators	PROBE

Figure 1-5 Workspace components

- Wire the circuit as shown below:

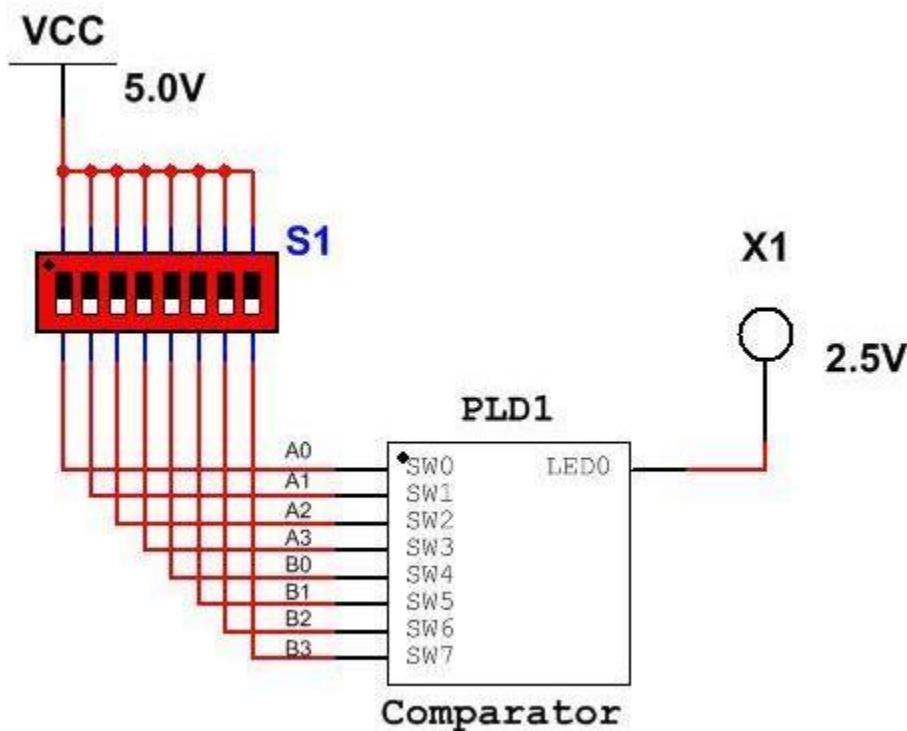


Figure 1-6 Circuit diagram

Part 3

Double-click the PLD sub circuit and select **Open sub sheet**.

- Select **Place > Component** and place the following components on the workspace:

Component	Group	Family
AND4	PLD Logic	LOGIC_GATES
XNOR2 (4 gates)	PLD Logic	LOGIC_GATES

Figure 1-7 Workspace components

Wire the circuit as shown below:

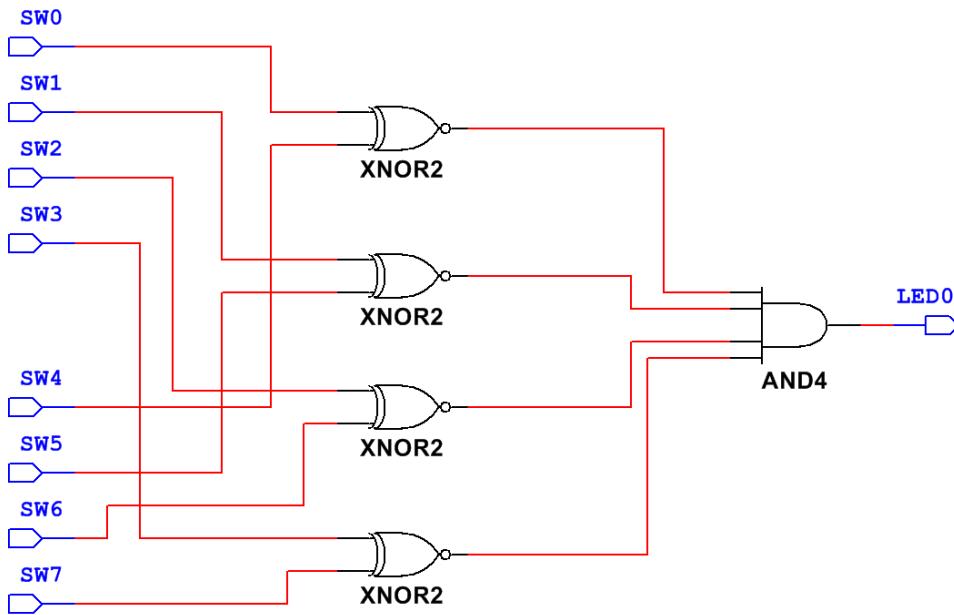


Figure 1-8 Circuit diagram

Deploying a 4-bit Parallel Comparator to the Digital Electronics Board

Return to the top-level circuit and run the simulation to verify the operation of the parallel binary comparator

- Use the switch pack to enter different values of binary numbers. The probe will illuminate when the inputs are exactly equal.

Note: Now we are ready to deploy the PLD schematic to the NI Digital Electronics Board. Connect the board to your computer.

- Double-click the PLD sub circuit and select **Open sub sheet**.
- Select **Transfer» Export to PLD**.
- Select **Program the connected PLD** and click **Next**.
- Click the **Refresh** button to make sure the Digital Electronics Board has been detected.
- Click **Finish**. Multisim will start the programming process.

Testing a 4-bit Parallel Comparator

After the PLD schematic has been successfully deployed to the Digital Electronics Board, experiment with the switches **SW0** to **SW7** to confirm the operation of the parallel binary comparator; observe the result in **LED0**.

- Record your observations below.

1.3 Conclusion

1-2 In this experiment we used a sub circuit. Sub circuits are used to ‘chunk’ larger, more sophisticated circuits into smaller, more manageable ones. How would the 4-bit parallel comparator circuit look on the PLD if it did not have a sub circuit?

1-3 What are the similarities and differences between a sub circuit and a chip component in Multisim?

1-4 What are the similarities and differences between comparators and adders?

1-5 Comparators are combinational logic circuits.

- A. True
- B. False

1-6 In a one bit magnitude comparator, the output A=B corresponds to which logic gate?

- A. NAND
- B. NOR
- C. XNOR
- D. NOT

1-7 The magnitude comparator begins by comparing the:

- A. Highest order bit
- B. Lowest order bit
- C. The bits that are equal
- D. None of the above

1-8 What combination of logic gates could you use to create a 4-bit comparator?

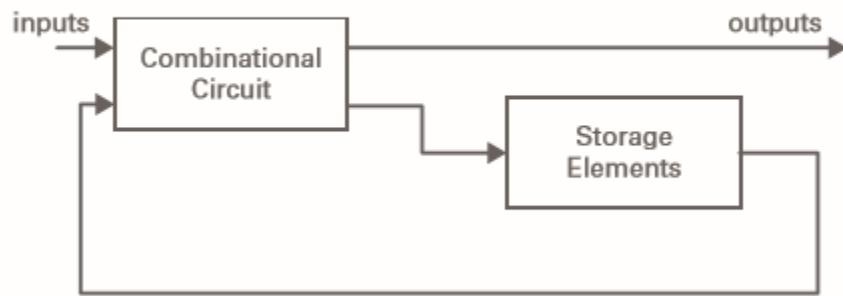
- A. 4 XNOR and 1 OR gate
- B. 4 XNOR and 1 NAND gate
- C. 4 XNOR and 1 NOT gate
- D. 4 XNOR and 1 AND gate

1-9 What is a sub circuit within the PLD design?

- A. It is a circuit that is a level below the main circuit
- B. It is a way to visually simplify a complex circuit
- C. It expands on a portion of the larger circuit
- D. All of the above

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 9: Latches and Sequential Logic Circuits

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 9: Latches and Sequential Logic Circuits

In all previous labs, we were dealing with combinational logic circuits. A combinational logic circuit is one such that all outputs may be determined from the current inputs. In this lab, we will introduce sequential logic circuits through the application of latches. A sequential logic circuit is one such that outputs depend on the previous inputs in addition to the current inputs. We will also look at how clock signals can be implemented into a circuit, building upon what was discussed briefly in Lab 7: Multiplexers and Demultiplexers.

Learning Objectives

In this lab, students will:

1. Understand the difference between synchronous and asynchronous sequential circuits.
2. Test and compare circuits for D latches using both logic gates and latches.
3. Confirm the characteristic table of a gated SR latch.
4. Observe and articulate the difference between D latches and SR latches.

Required Tools and Technology

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Expected Deliverables

In this lab you will collect the following deliverables:

- Latch features table
- Latch comparisons
- Observations of probe behavior
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

Sequential Circuit

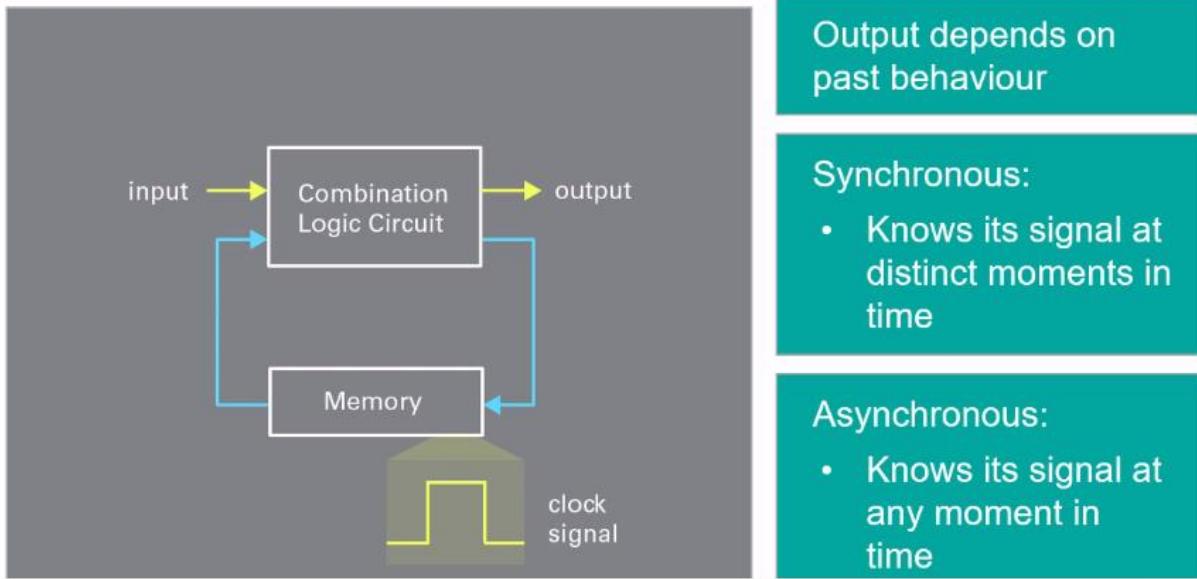


Figure 1-1 Video. View the video here: <https://youtu.be/RMY4rTyVso0>



Video Summary

- A sequential circuit's output depends on the present combination of inputs, and the past behavior of the circuit
- There are two classes of sequential circuits: synchronous and asynchronous
- The clock signal is a rectangular pulse train

Sequential Circuit

A *sequential circuit* is one whose output depends not only on the present combination of the inputs, but also on the past behavior of the circuit.

- The basic building block of sequential circuits is the bistable, a circuit having two stable states.
- The state of a sequential circuit is represented by a set of bits, called state variables, containing all the information about the past necessary to explain the future behavior of the circuit.
- The outputs of sequential circuits are a function of both their inputs and of the history of these inputs.

The block diagram of a sequential logic circuit is presented below.

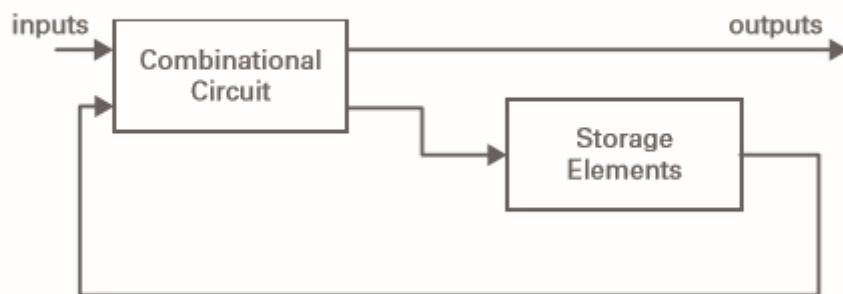


Figure 1-2 Sequential logic circuit block diagram

There are two main classes of sequential circuits, depending on the timing of their signals:

1. *Synchronous sequential circuits* – their behavior can be defined knowing its signals at distinct moments in time.
2. *Asynchronous sequential circuits* – their behavior can be defined knowing the input signals at any moment in time and the order in which they change.

This lab presents synchronous sequential circuits, which employ a block signal for synchronization. Throughout this lab, synchronous sequential circuits will be simply called sequential circuits.

Clock Signals

The clock signal is a rectangular pulse train. It has a precise pulse width and a precise interval between pulses, called clock cycle time.

- We encountered these briefly in *Lab 7: Multiplexers and Demultiplexers*.
- These signals are used by sequential circuits by synchronizing the activity within the circuit and for the synchronizing the update of stored values.
- Most sequential circuits change their state on one of the edges of the clock pulses (rising or falling edge), being referred to as *edge-triggered*.
- The timing events is important when dealing with sequential logic circuits. Therefore, in this case, besides the truth table, one should also know the order in which events occurred, i.e. the *logic timing diagram*.
- The table presenting the sequential circuits operation is often called a *characteristic table* rather than a truth table, since it does not represent a combinational circuit.
- The majority of digital systems are principally synchronous circuits are easier to design and troubleshoot, changing outputs only at specific moments in time. They always contain some asynchronous circuits too.
- An example pulse train is shown below (left), as well as an example timing diagram for an AND gate (right).

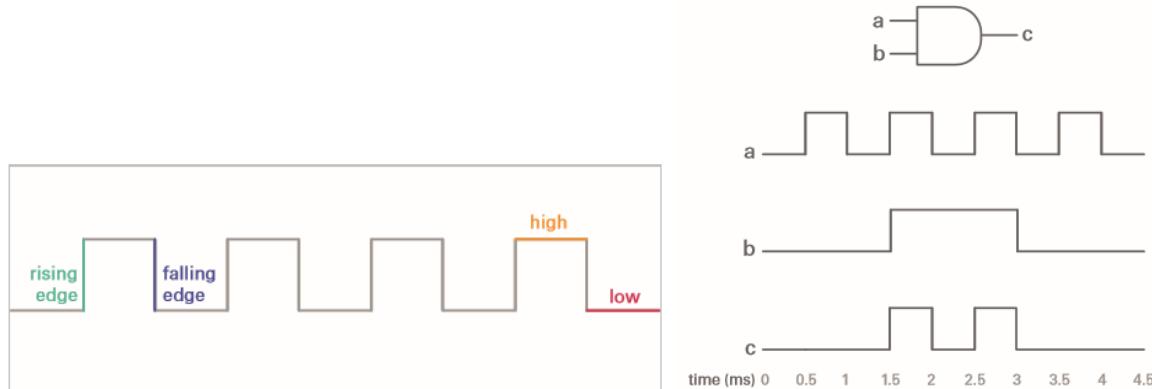


Figure 1-3 Pulse train (left) and timing diagram for an AND gate (right)

The SR Latch

A *latch* is a storage element that operates with signal levels rather than signal transitions.

- A latch is a level sensitive device and the basic building block of flip-flops, which will be presented later.
- One of the simplest sequential circuits is the basic latch.

The basic latch is presented in the figure below and has the functionality described by the accompanying characteristic table.

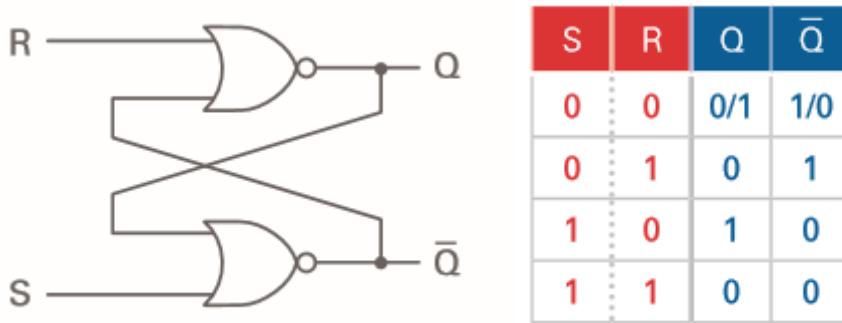


Figure 1-4 Basic latch (left) and characteristic table (right)

- The circuit has two inputs, S (set) and R (Reset), and two outputs, Q and \bar{Q} , with \bar{Q} the complement of Q.
- The state of the latch can be controlled through the S and R inputs, which set and reset the output Q.
- The word “set” denotes the process of making the output Q a logic value 1, while “reset” denotes the process of making the output Q a logic value 0.

Note: The output of the circuit depends not only on the current inputs, but also on the previous value of the output.

- When both the inputs are 0, the output remains unchanged.
- A pulse on the S input sets the output to 1 and a pulse to R resets the latch ($Q=0$)
- Both S and R set to 1 does not represent a valid combination of the inputs because this force the two outputs to 0, resulting in an unstable circuit.
 - In practical applications, setting both inputs to 1 is forbidden.

The symbols for a SR latch are presented in the figure below



Figure 1-5 SR latch symbols

The SR latch is bistable element with one bit of state stored in Q. Its state can be controlled through the two inputs, S and R. When none of the two inputs is asserted, the state of the circuit remains unchanged. The SR latch represents the basic element for most static memory structures.

The Gated SR Latch and D Latch

The *SR latch* changes its states at random moments in time, when the outputs are changed.

- Its operation can be modified in such a way that an input signal (*ENABLE*) controls the moment when the state of the latch changes.
- The circuit is called a gated SR latch.
- The *ENABLE* signal can be used an ON/OFF signal, a synchronizing signal, or a clock signal.
- The characteristic table (center) presents the operation of the gated SR latch. $Q(t)$ represents the current value of the output and $Q(t+1)$ represents the next state.
- The SR latch circuit is shown below (left) along with its graphical symbol (right).

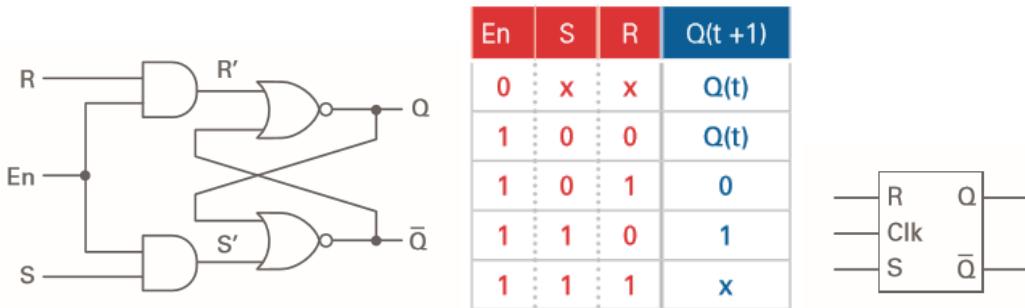


Figure 1-6 latch circuit (left), characteristic table (center), and graphical symbol (right)

The *D latch* eliminates the undesirable condition of the indeterminate state in the SR latch.

- It ensures that the inputs S and R are never equal to 1 in the same time.
- The circuit has only two inputs, D and Clk or En.
- The D input stands for data.
- The D latch circuit is presented in the figure below (left), along with the D latch characteristic table (center) and graphical symbol (right).



Figure 1-7 D latch circuit (left), latch characteristic table (center), and graphical symbol (right)

1-1 How are sequential circuits different from combinational logic circuits?

1-2 What is the difference between synchronous and asynchronous sequential circuits?

1-3 Fill out the table below highlighting the distinct features of each of the following latches.

SR	GATED SR	D

1.2 Implement: Building a Gated D Latch Circuit using Logic Gates

D Latch Circuit 1

Launch Multisim.

- Build the following D latch circuit using logic gates:

Note: U2 is a digital constant set a **high** and S1 is a button **PB_NO**.

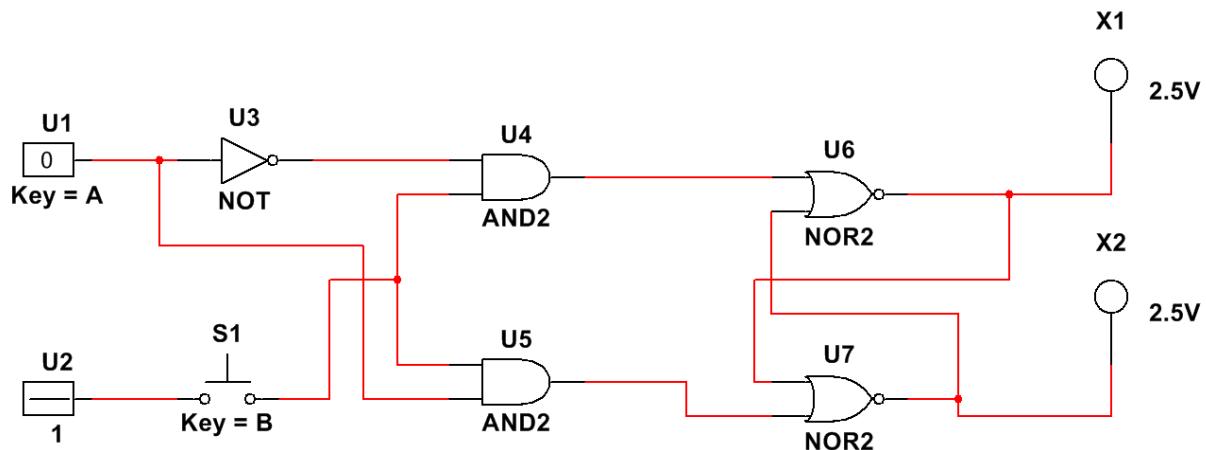


Figure 1-8 D latch circuit diagram

Testing a Gated D Latch Circuit using Logic Gates

- Start the Simulation.
- Change U1 to **1**.

1-4 Do the probes change and why?

Note: Our clock signal is being simulated by **S1**, an interactive button.

- Press the button.

1-5 Does the probe change and why?

- Change U1 to **0**.

1-6 Do the probes change and why?

- Press the button.

1-7 Does the probe change and why?

- Stop the simulation.

1-8 Compare your observations to the second characteristic table in Step 3.

1-9 According to your observations, explain how the D latch works.

Exploring D Latch Behavior Circuit 2

- Launch Multisim.
- Connect the following circuit:

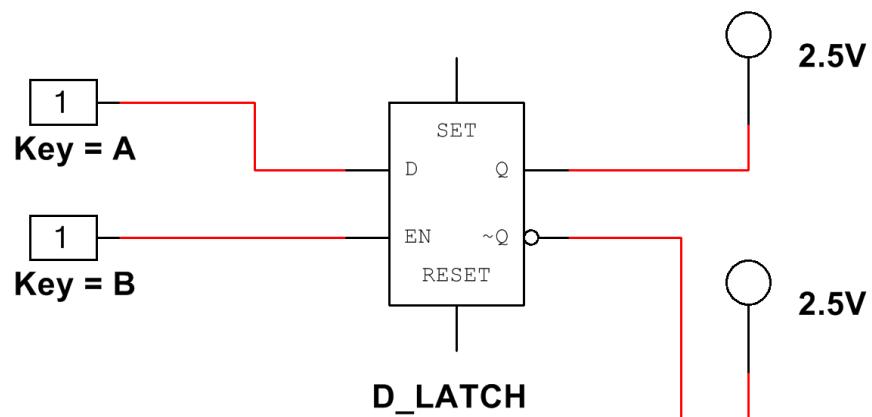


Figure 1-9 D latch circuit diagram

- Run the circuit.
- Keep the Enable input at **0** and vary the D input by pressing the **A** key on your keyboard.

1-10 What happens to the probes?

- Keep the Enable input at **1** and vary the D input by pressing the **A** key on your keyboard.

1-11 What happens to the probes?

1-12 Is this behavior of Circuit 2 the same as that of Circuit 1?

1.3 Exercise: Verifying the Gated SR Latch Characteristic Table

Circuit

- Launch Multisim
- Connect the following circuit:

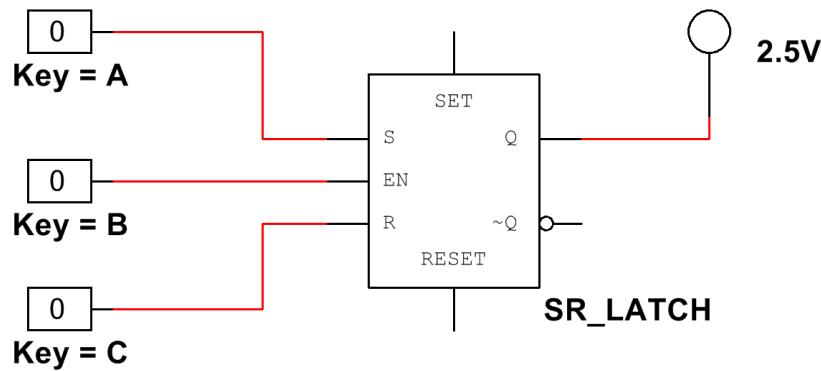


Figure 1-10 SR latch circuit diagram

- Start the simulation.
- Using the Characteristic table provided, vary the inputs and observe the probe.

En	S	R	Q(t + 1)
0	x	x	Q(t)
0	0	0	Q(t)
1	0	1	0
1	1	0	1
1	1	1	x

Figure 1-11 Characteristic table

1-13 With the Enable input equal to 0, does it matter what you do to the S and R inputs?

- Set Enable to **1**.
- Set S to **1**.

1-14 Does the probe light up?

- A. Yes
- B. No

- Set S to **0**.

1-15 Does the probe light up?

- A. Yes
- B. No

1-16 Explain what has happened.

- Set R to **1**.

1-17 Does the probe light up?

- A. Yes
- B. No

1-18 Explain what has happened.

- **Stop** the simulation.

1-19 Were you able to observe the indeterminate states on the characteristic table?
When S and R are both equal to 1, what does the probe show?

1.4 Conclusion

1-20 Based on your observations, what are the main differences between the behavior of D latches and SR latches?

1-21 When would a D latch be useful? An SR latch?

1-22 What is the difference between the clock (Clk) input and the enable input (En)?

1-23 A sequential circuit's behavior depends on:

- A. The number and type of latches
- B. Only on the inputs
- C. Current inputs and previous behavior
- D. The previous 4 bits

1-24 Sequential circuits employ which component for synchronization?

- A. SR latches
- B. Clock signals
- C. Gated D latch
- D. None of the above

1-25 A gated SR latch is a combination of which logic gates?

- A. 2 AND and 2 NOR gates
- B. 2 AND and XNOR gates
- C. 2 AND and XOR gates
- D. 2 AND and NOT gates

1-26 In the characteristic table of a gated SR latch, Q (t+1) represents:

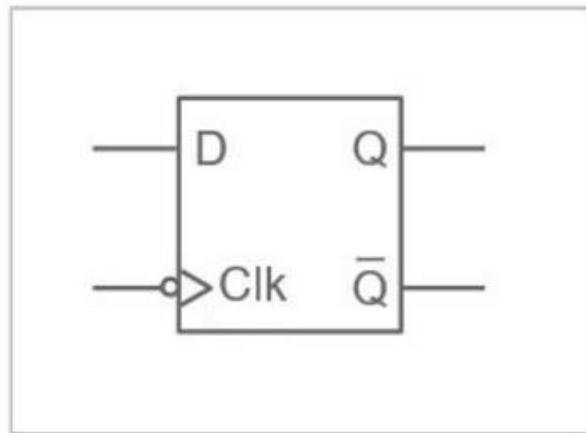
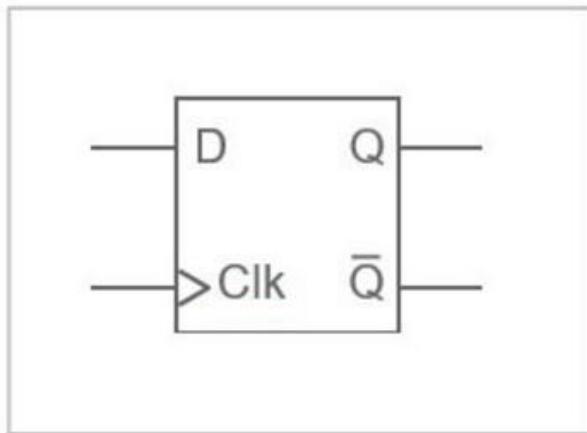
- A. A don't care condition
- B. The current value of the output
- C. Removing the Q (t+1) state
- D. Adding another output

1-27 The D latch eliminates the undesirable condition of the indeterminate state in the SR latch by:

- A. Reducing the number of logic gates needed for the circuit
- B. Having only the following two inputs: D and Clk or En
- C. Removing the Q(t + 1) state
- D. Adding another output

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 10: Flip-Flops

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 10: Flip-Flops

In the previous lab, you were introduced to the concept of sequential logic circuits by examining latches. Particularly, you explored D and SR latches. In this lab, you will explore sequential logic circuits by focusing on flip-flops. Flip flops differ from latches in that they are edge triggered. Latches are level sensitive devices. Flip-flops can be constructed from latches. Together, flip-flops and latches are the building blocks of sequential logic circuits.

Learning Objectives

In this lab, students will:

1. Become familiar with the basic behavior of D, JK and T flip-flops
2. Explore variations of flip-flops in greater detail (i.e. master-slave relationships)
3. Gain an awareness of the versatility of the DFF and its application in other circuits.

Required Tools and Technology

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
[http://www.ni.com/gate/gb/GB
ACADEMICEVALMULTISIM/
US](http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US)
 - ✓ View Help:
[http://www.ni.com/multisim/tec
hnical-resources/](http://www.ni.com/multisim/technical-resources/)
-

Expected Deliverables

In this lab, you will collect the following deliverables:

- Flip-flop analysis
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

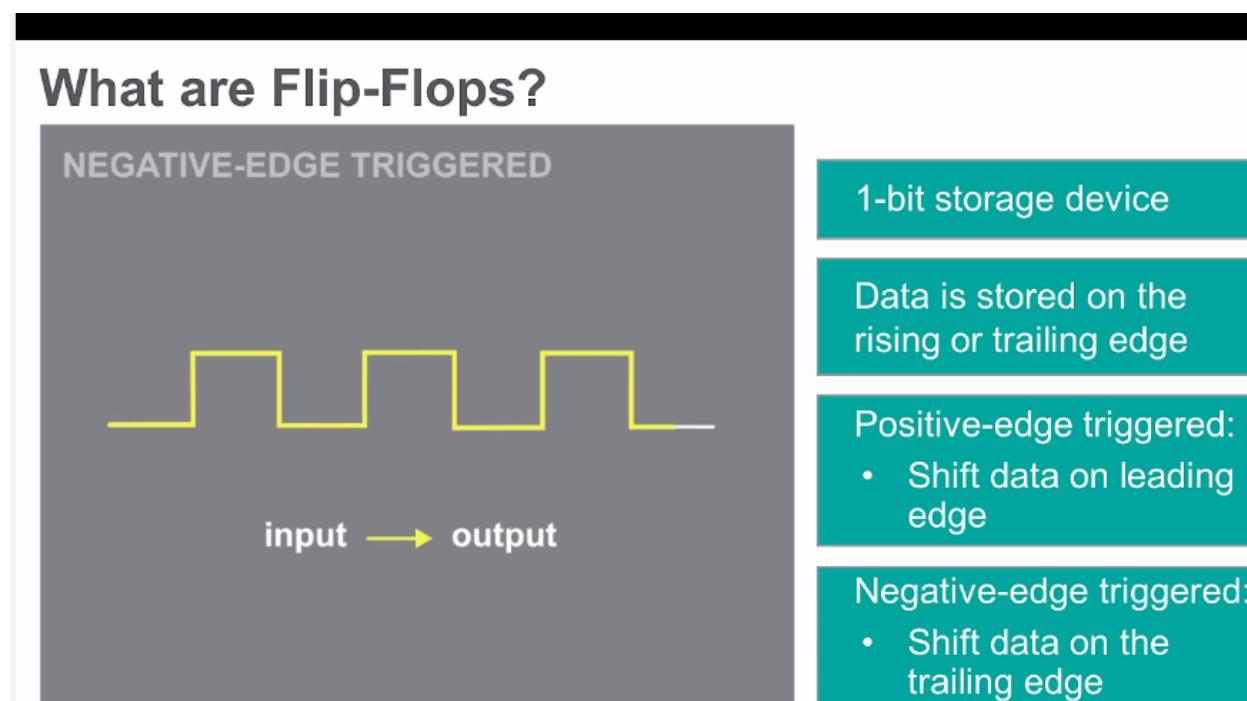


Figure 1-1 Video Screenshot. View the video here: <https://youtu.be/zLZUjC5Zr-w>



Video Summary

- Flip-flops are 1-bit storage devices that store data on the falling or rising edge of a clock signal
- If a signal is transferred from input to output on the rising edge it is positive-edge triggered
- If a signal is transferred from input to output on the falling edge, it is negative-edge triggered

Flip-flops

- Flip-flops are the fundamental building blocks of sequential circuits.
- They are 1-bit storage devices that, unlike latches (which are level sensitive devices) are edge-triggered.
- Data gets stored into a flip-flop at one of the edges of the clock signal, i.e. when the clock input makes a transition from 0 to 1 or from 1 to 0.

The timing diagram of a clock pulse is shown below:

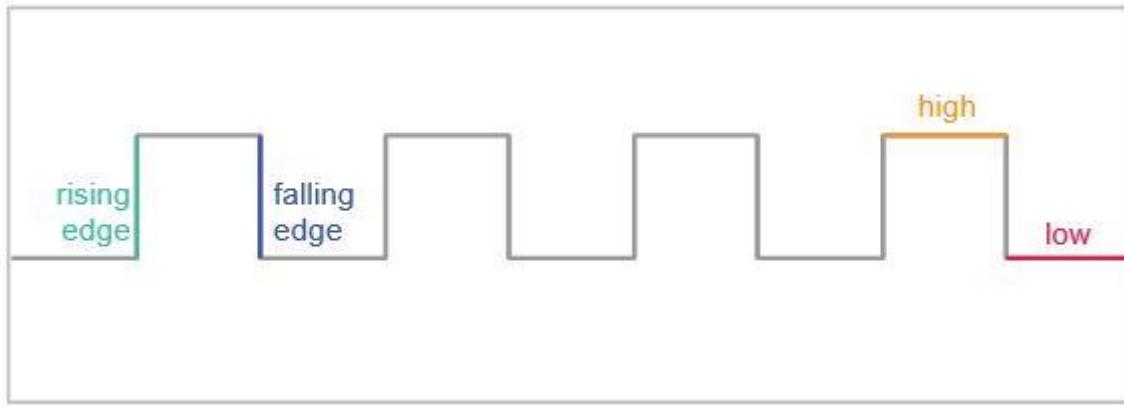


Figure 1-2 Timing diagram of a clock pulse

Latch Feedback

- The state transitions of the latches start in the moment the clock changes to 1 and can continue for the entire period while it is active.
- Because of this, the output of a latch cannot be applied through combinational circuits to the inputs of other latches triggered by the same clock signal.
- This fact represents a serious drawback when using latches as storage elements.
- Flip-flops overcome this problem by triggering only during signal conditions.

Latch feedback is illustrated in the figure below.

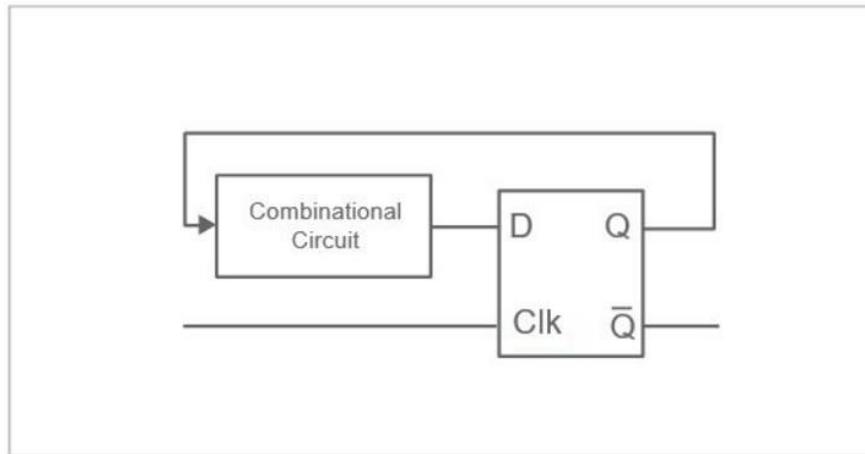


Figure 1-3 Latch feedback

D Flip-flops (DFFs)

D flip-flops (DFFs) are very commonly used because they are perfectly suited for the construction of sequential circuits.

- If the DFF transfers the input to the output on the rising edge of the clock signal then it is said to be a *positive-edge triggered* D flip-flop or positive-edge DFF.
- If the DFF transfers the input to the output on the falling edge of the clock signal then it is said to be a *negative-edge triggered* D flip-flop or negative-edge DFF.
- This mode of operation is signaled through the use of a circle on the clock input. The image below shows the symbol for a positive-edge DFF (left) and a negative-edge DFF (right).
- For both DFF circuits shown below, the small triangle on the clock input indicates that the flip-flop will trigger only on one of the clock edges.
- The complemented output, \bar{Q} is omitted in the graphical symbol when it is not needed.

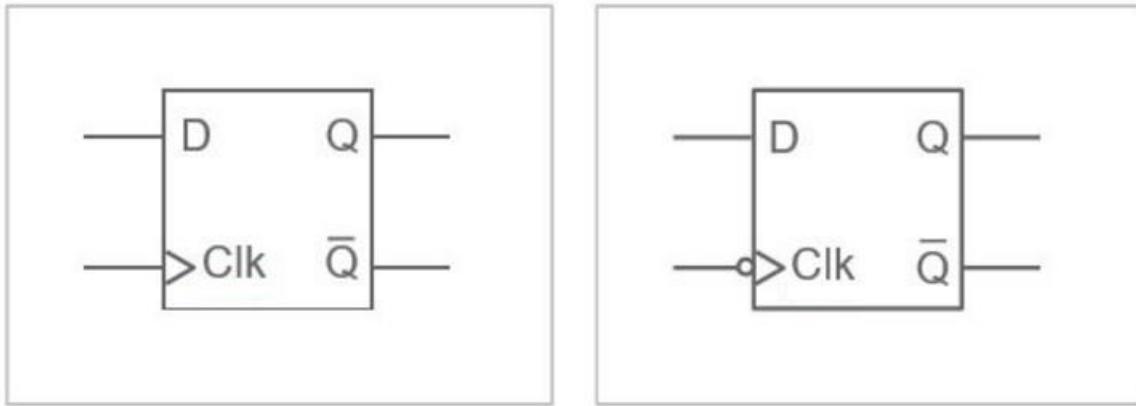


Figure 1-4 Positive edge DFF (left) and negative edge DFF (right)

The operation of the positive-triggered D flip-flop is very simple:

- The output **Q** will go to the present state of the **D** input when the clock signal changes from **0** to **1**.
- The level present at the input will be stored in the flip-flop on the rising transition of the **Clk** signal.

The figures below present the functionality of a positive-edge DFF:

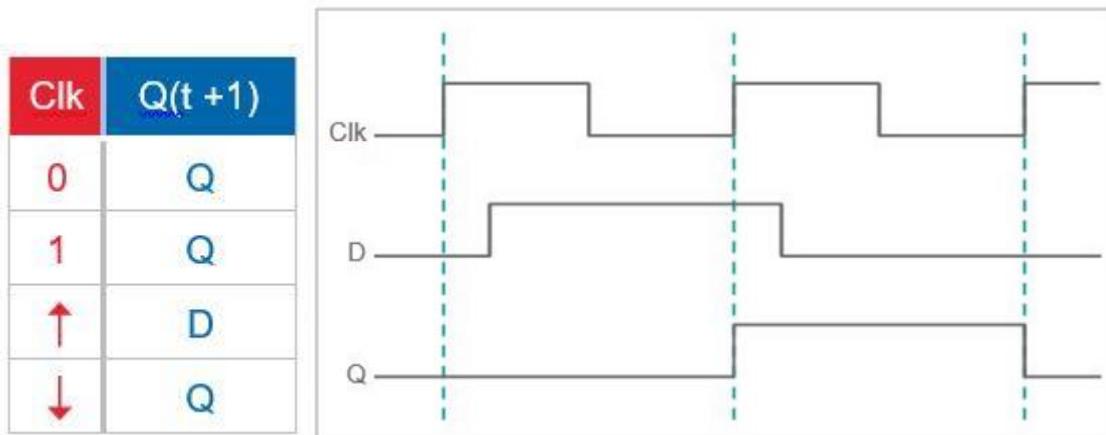


Figure 1-5 Characteristic table (left) and diagram (right) for a positive-edge DFF

A flip-flop can be built using the following methods:

- Using two D latches arranged in a master-slave configuration, where one of them is transparent during one semi-period of the clock period and the other transparent during the other semi-period.

Note: another circuit having the same operation, but implemented using fewer gates, can be built from three SR latches.

- Using of a single D latch and short pulses derived from the clock signal, which cause of the latch to be transparent for only a brief moment. These circuits are called pulsed latches or pulse-based flip-flops.

The timing characteristics of flip-flops include:

- the propagation delay from **Clock** to **D** (the time needed for a value present on the input to reach the output when the clock changes from **0** to **1**)
- the setup time
- the hold time
- the maximum frequency that can be applied to the clock signal and still have correct operation

The J K Flip-flop (JKFF)

Another type of flip-flop, one that is not as widely used as the DFF, but a more versatile one, is the *J K flip-flop (JKFF)*.

- The J K flip-flop is referred to as the *universal* bi-stable because it can easily behave like any one of the others.
- It has two data inputs, (**J** and **K**), a clock input, and two outputs (**Q** and \bar{Q})
- **J** acts as a **SET** input and **K** acts as a **RESET** input.
- The output changes only on one of the clock edges.
- When both **J** and **K** are **1**, the flip-flop toggles between opposite logic states each time the clock signal changes from **0** to **1** from **1** to **0**.
- This circuit is like the one of a gated SR latch with an edge detector, with the difference that the *forbidden* state (when both **S** and **R** are set to **1**) is replaced by the *toggle* state.

The golden rules of JKFFs are:

1. If **J** and **K** are different, then **Q** is always the same as **J**.
2. If **J** and **K** are **0**, nothing happens.
3. If **J** and **K** are **1**, **Q** toggles.

The figures below present the way in which a J K flip-flop can be built using a DFF and logic gates. The graphical symbol of the JKFF is also shown below, and its operation is described by the characteristic table.

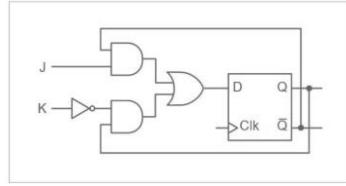


Figure 1-6 J K flip-flop built using a DFF and logic gates

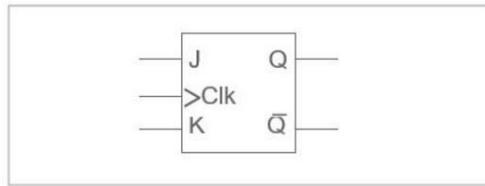


Figure 1-7 Graphical symbol of the JKFF

J	K	Q(t +1)
		Q(t)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q(t)

no change
reset
set
toggle

Figure 1-8 JKFF characteristic table

The T flip-flop

A *T flip-flop* is a flip-flop whose output toggles between **1** and **0** each applied clock pulse, when an input called **T** (from toggle) is active.

- The toggle mode of operation of a flip-flop is sometimes useful in synchronous circuits.
- It can be implemented using a DFF.

The figures below show the CLC for a TFF built using a DFF. The TFF graphical symbol is also shown below, along with the TFF characteristic table.

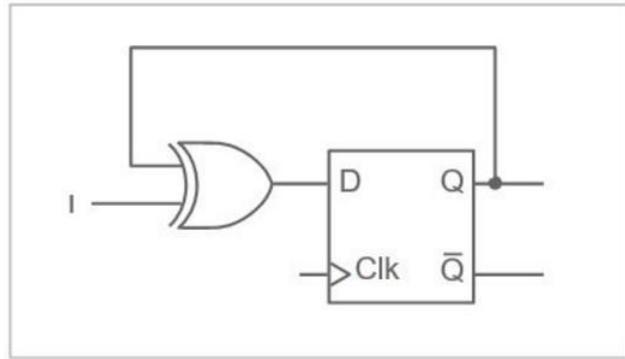


Figure 1-9 CLC for a TFF built using a DFF

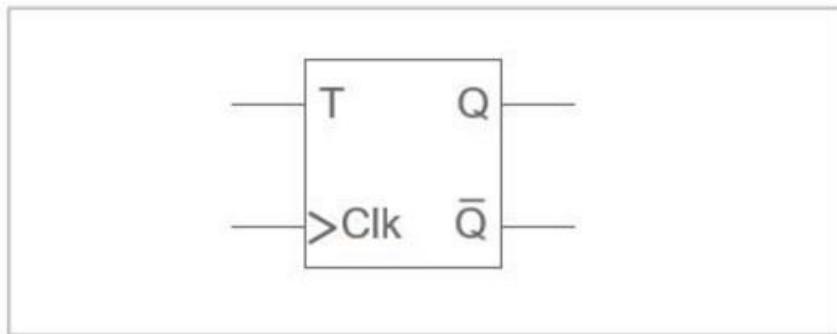


Figure 1-10 TFF graphical symbol

T	Q(t + 1)
0	Q(t)
1	Q(t)

no change

toggle

Figure 1-11 TFF characteristic table

A T flip-flop can also be built using a J K flip-flop, having the inputs **J** and **K** tied together, as shown below.

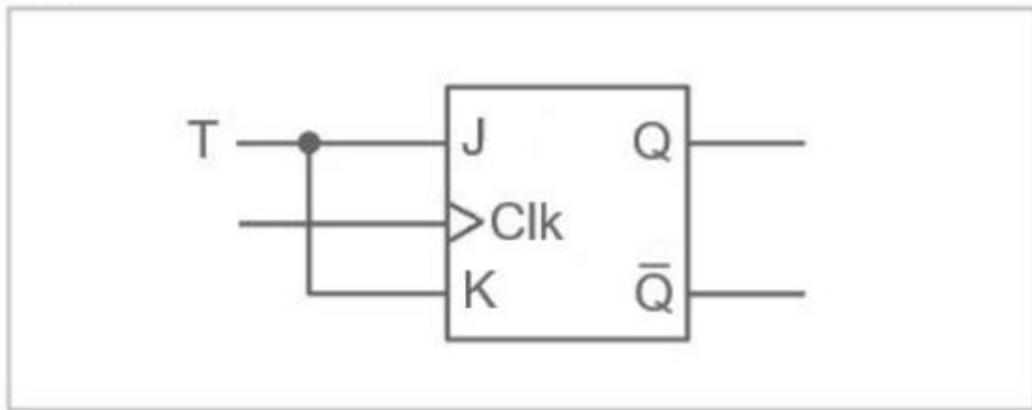


Figure 1-12 TFF built using a JKFF

1-1 Flip-flops are edge-triggered. What does this allow you do that latches couldn't?

1-2 Comparing the D Flip-Flop and the JK Flip-Flop, which do you believe to be the most practical? Why?

1.2 Simulate: Building a Master-Slave D Flip-Flop

Instructions:

- Launch Multisim
- Connect the following circuit:

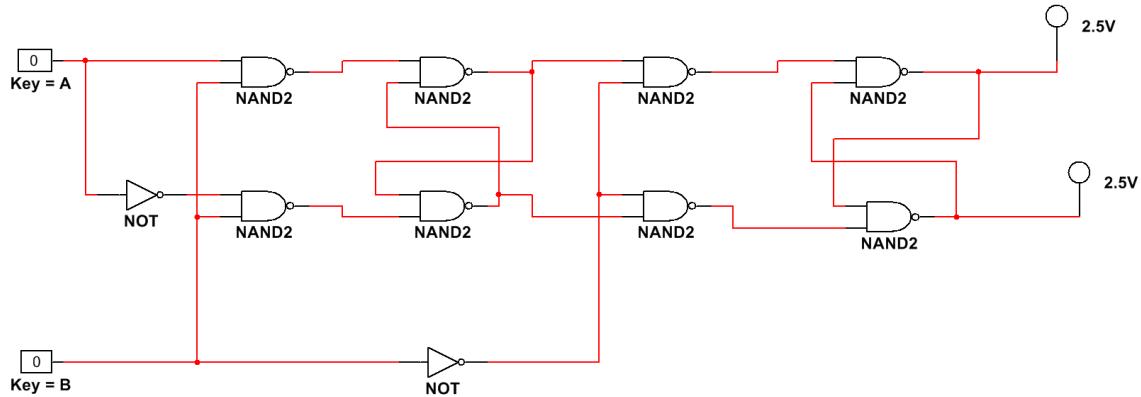


Figure 1-13 Circuit diagram

- Start the simulation
- Set the top input to 1

1-3 Does the probe change value?

- A. Yes
B. No

- Set the bottom input to 1

1-4 Does the probe change value?

- A. Yes
B. No

- Set the bottom input to 0

1-5 Does the probe change value?

- A. Yes
 B. No

1-6 Does toggling the top input change the values of the probes?

- A. Yes
 B. No

1-7 When the bottom input changes, the probes change to match the top input:

- A. Always
 B. Sometimes
 C. Never

1-8 Is this circuit positive edge triggered or negative edge triggered?

- A. Positive edge triggered
 B. Negative edge triggered

- **Stop** the simulation when you are done

1.3 Simulate: Building a JK Flip Flop Using a DFF and Logic Gates

- Launch Multisim
- Connect the following circuit:

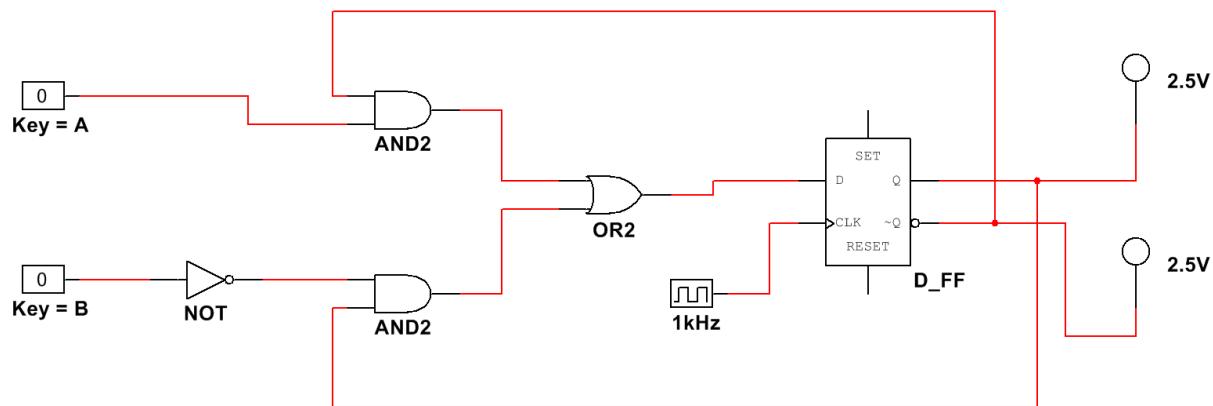


Figure 1-14 Circuit diagram

- Start the simulation
- Vary the two inputs and observe what happens

1-9 Compare your results with the following characteristic table:

J	K	Q(t +1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q(t)

Figure 1-15 JKJK characteristic table

- **Stop** the simulation when you are done

1-10 Based on the behavior, which interactive input is the J input and which is the K input?

1.4 Simulate: Building a T Flip-Flop

- Launch Multisim
- Connect the following circuit:

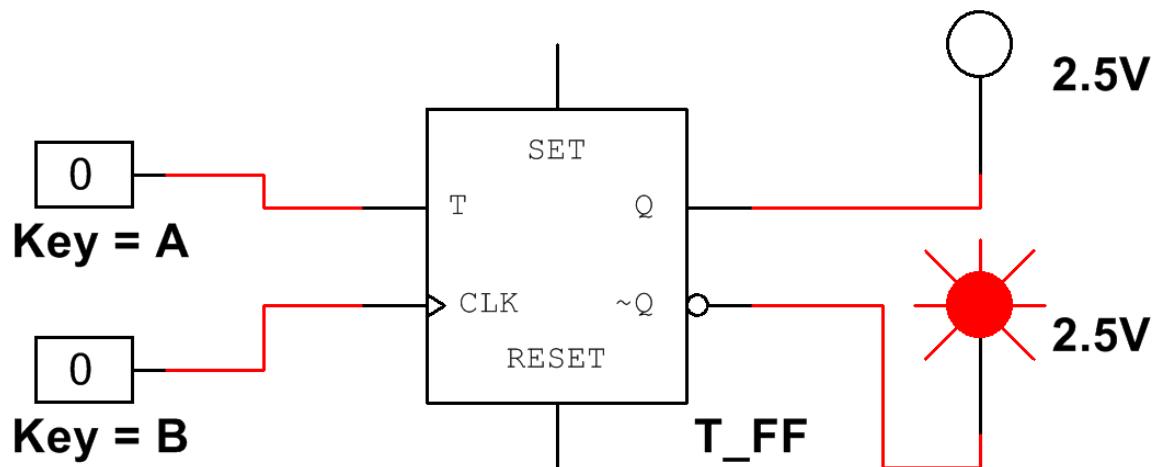


Figure 1-16 Circuit diagram

- Start the simulation
- Vary the input and observe the probes

1-11 What happens when the input is high?

1-12 What happens when the input is low?

1-13 How is this different than the other flip-flops?

- **Stop** the simulation when you are done

1.5 Conclusion

1-14 Why is it useful to be able to simulate circuits with both logic gates and chips when they behave the same way?

1-15 In the first part of this experiment, the flip-flop was triggered on the falling edge. Use this information to explain how the probes in this part lit up.

1-16 When might you use a "T" type flip-flop?

1-17 Flip-flops are storage devices that:

- A. Are level sensitive
- B. Are edge-triggered
- C. Work only when the clock signal is set to 1
- D. Work only when the clock signal is set to 0

1-18 Some D flip-flops have a circle on the clock input. This indicates:

- A. A negative flip-flop
- B. The current travels into the flip-flop in one direction
- C. The flip-flop will be triggered only on one of the clock edges
- D. The Q output is equivalent to the D input

1-19 JK flip-flops are said to be versatile because:

- A. They have two inputs, 'set' and 'reset'
- B. They can behave like any other flip-flop
- C. The output changes only one of the clock edges
- D. All of the above

1-20 If both inputs on a JK flip-flop are 1, then:

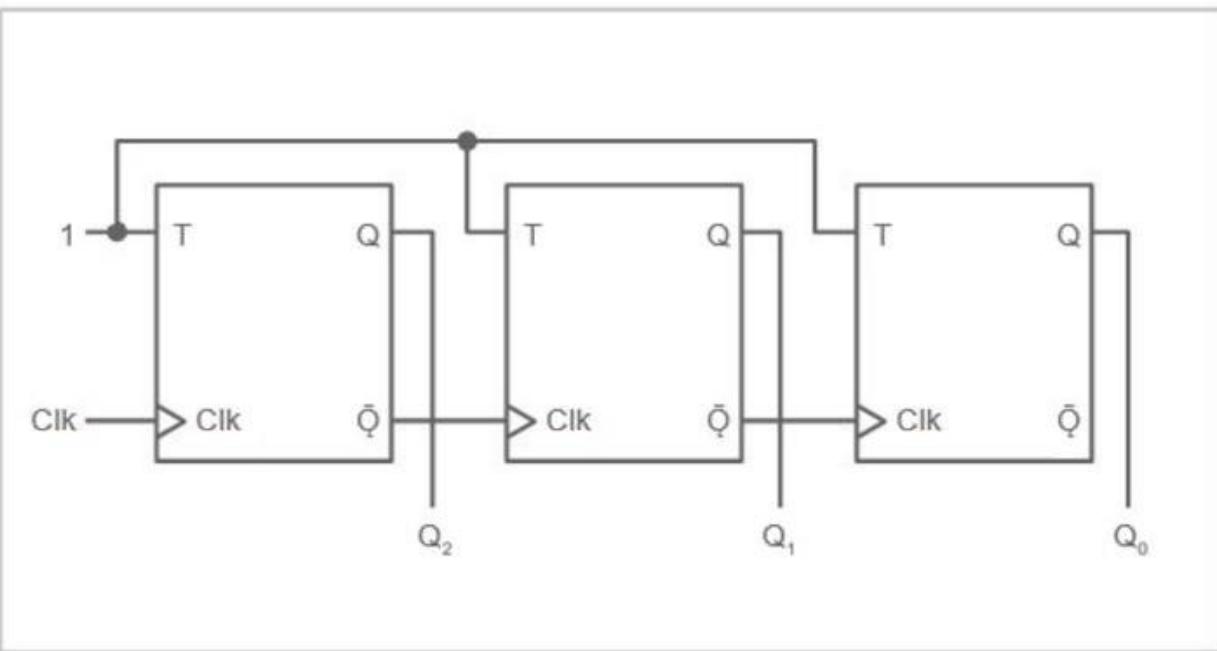
- A. The Q output is always the same as the set input
- B. The Q output is always the same as the reset input
- C. Q toggles
- D. Nothing happens

1-21 T flip-flops can be built with which of the following:

- A. JK flip-flop and an OR gate
- B. D flip-flop and an XOR gate
- C. JK flip-flop and an XOR gate
- D. D flip-flop and an OR gate

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 11: Counters

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 11: Counters

Counters are sequential arithmetic circuits used, as the name suggests, for counting the number of input pulses. The counter goes through a predetermined sequence of states as the clock pulses, and if it is a straight binary counter these reflect the binary number sequence. They can be used for counting event occurrences, for generating timing intervals for the control of various tasks, for measuring the time elapsed between specific events, and so on.

Counters come in two groups:

1. *Asynchronous counters* (ripple counters), where the output of one flip-flop is connected to the clock input of the next one.
2. *Synchronous counters*, where the clock signal is connected to the clock input of all the flip-flops.

Learning Objectives

In this lab, students will:

1. Become familiar with the differences between synchronous and asynchronous counters.
2. Observe theoretically and practically that flip-flops are the building blocks of counters.
3. Explore the workings of different types of counters and their practical applications using Multisim and the Digital Electronics Board.

Required Tools and Technology

Platform: NI ELVIS III

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
 - ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8zialSxv0gwtshBA2dh_M
-

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>
-

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
 - ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>
-

Expected Deliverables

In this lab, you will collect the following deliverables:

- Circuit diagram of a three-bit binary ripple counter
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

What are Counters?

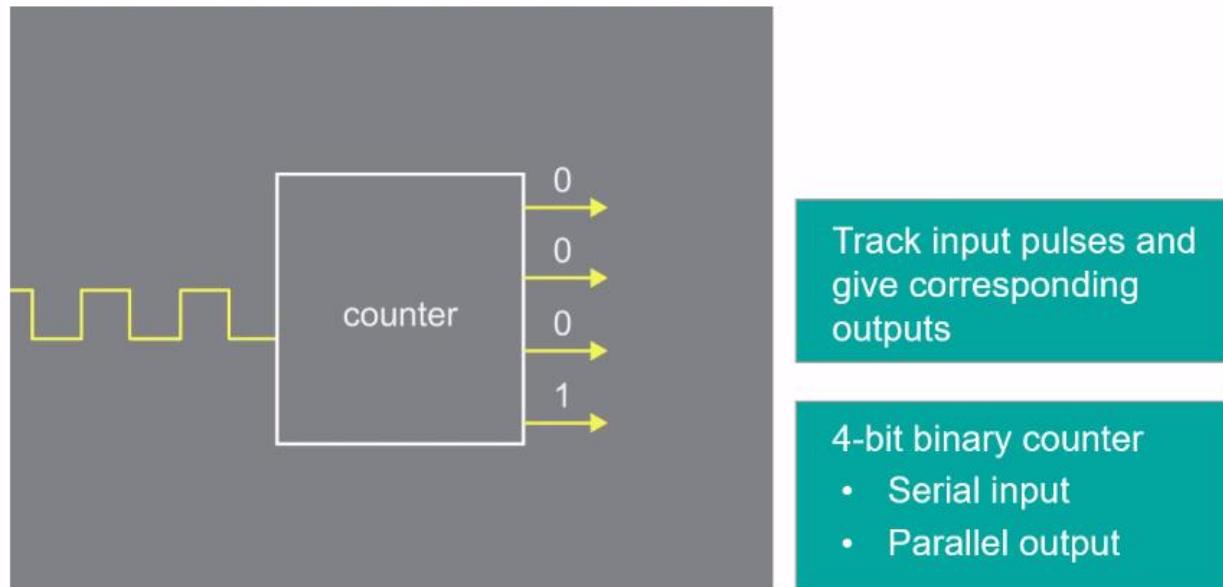


Figure 1-1 Video Screenshot. View the video here: <https://youtu.be/7sgHyAX9qN8>



Video Summary

- Counters track the number of input pulses and give an output that corresponds to it
- In a 4-bit counter, one pulse input to the counter gives an output of 0001 and the second would be 0010, etc.
- The maximum number of bits that the counter has to track determines how many flip-flops will be needed for the circuit

Asynchronous Counters

These counters are named this way because the flip-flops do not change states in exact synchronism with the applied clock signal, with only the first one responding to the clock pulses.

- The figure below presents the simplest circuit for a three-bit up-counter, built using T flip-flops.
 - The T input of each one of the three flip-flops is connected to 1, so they will toggle each time the clock input changes its state from 0 to 1.
 - These types of counters are also called *ripple counters* because their behavior is similar to that of the rippling carries in a ripple-carry adder.
 - They can also be implemented using D or JK flip-flops.

In any counter, the signal on the output of the last flip-flop will have a frequency having the value of the input clock frequency divided by the mod number of the counter. The major drawback of the asynchronous counters comes from the effects of the propagation delay in the flip-flops, making these types of circuits useful only in low-frequency applications.

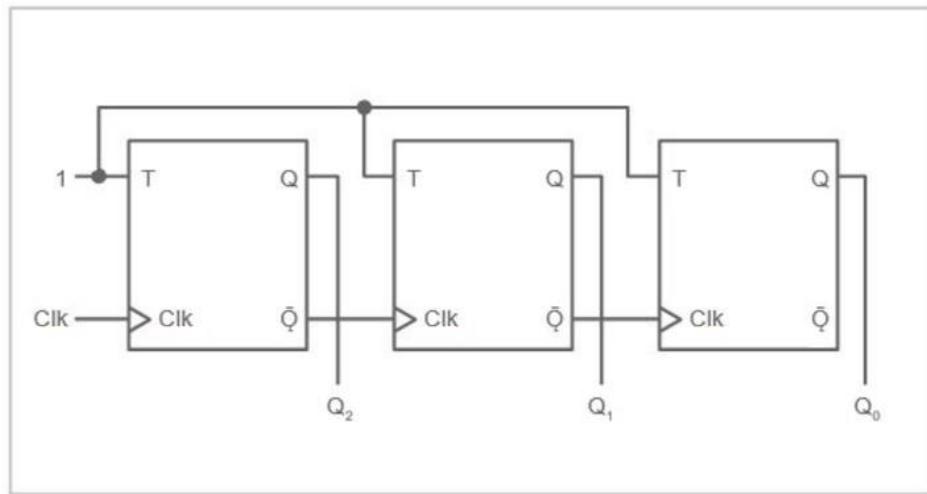


Figure 1-2 Asynchronous Counters

Synchronous Counters

Synchronous counters have the clock signal applied directly to the clock input of all the flip-flops composing them. All the flip-flops being triggered simultaneously (in parallel), these counters are also named *parallel counters*. They overcome the problem of the accumulating delay in the ripple counters by applying the clock pulses to all the flip-flops. The decision whether one of them is to complement its output is determined from the values of the data inputs, such as **T** or **J** and **K**, at the time of the clock edge.

In the case of a binary counter, the flip-flop corresponding to the least significant bit is complemented with every clock pulse. Any other flip-flop will toggle when all the outputs of the flip-flops in the lower significant stages are **1**. The synchronous counter can update its state on the positive or on the negative edge of the clock signal and can be built using **J K** flip-flops, **T** flip-flops or **D** type flip-flops with XOR gates.

- The parallel counter eliminates the accumulating delays in the ripple counters and can operate at high frequencies, with the cost of increased power consumption and of a more complex circuit.
- The counter below is a synchronous counter with serial enable, having the toggle-enable signals computed serially.
 - Another way of building the counter is by placing AND gates of increased size for each stage, something that could lead to fan-in problems.
 - In this last case, the counter is said to be a synchronous counter with parallel enable.

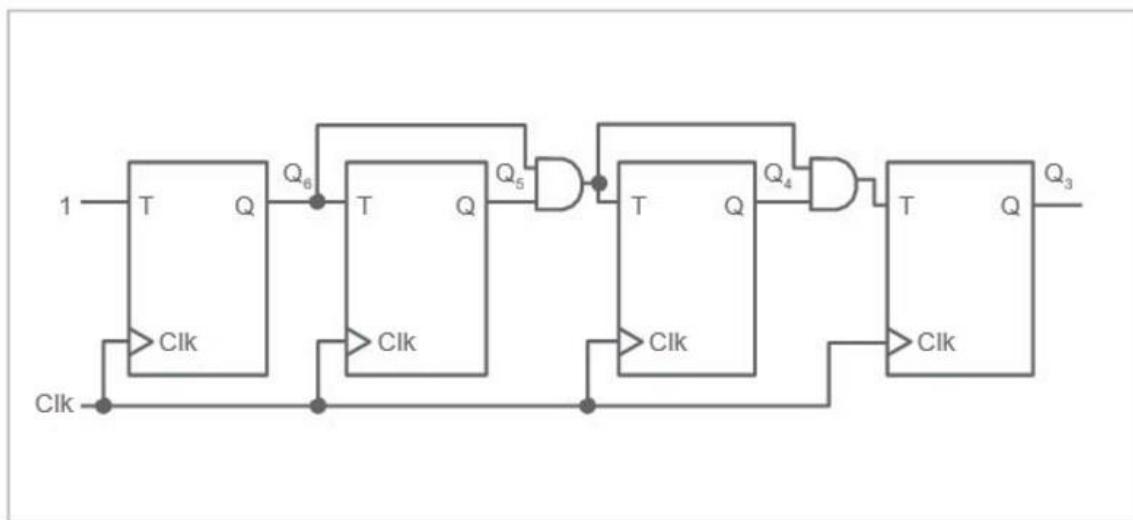


Figure 1-3 Synchronous Counters

Enable and Clear Inputs

- The *Enable* input inhibits counting no matter the clock signal states.
- The asynchronous resetting of the counter may also be required in some cases.
- A low level at the *Clear* input sets all flip-flops in the circuit at low levels, regardless of input.
- The following figure shows a four-bit synchronous up-counter with enable and clear inputs:

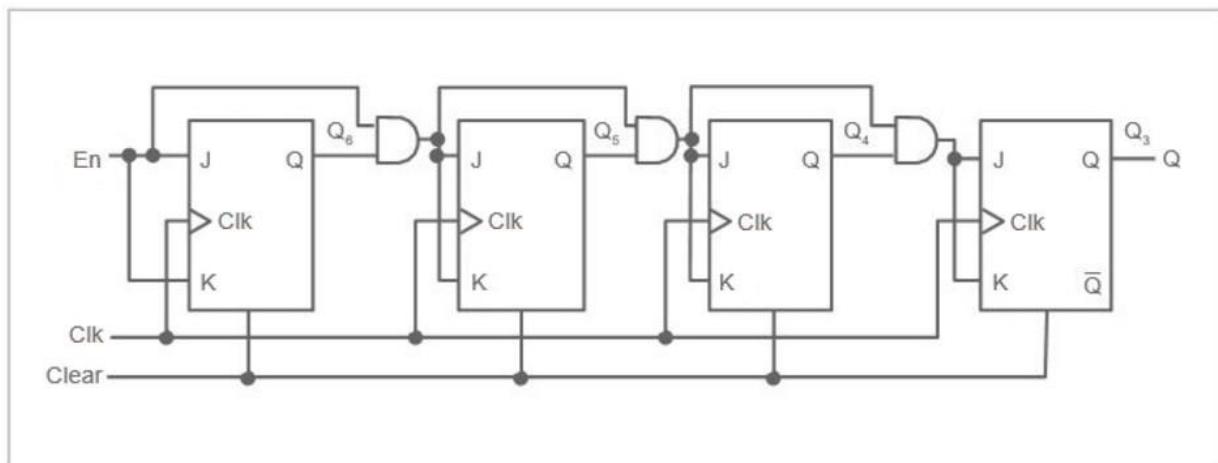


Figure 1-4 Enable and clear inputs

1-1 What is the difference between Enable and Clear inputs and how do they affect the function of a counter?

- Draw a circuit diagram of a three-bit binary ripple counter using only logic gates. Include the diagram with your completed lab.

1.2 Simulate: Building a Three-bit Binary Ripple Counter

- Launch Multisim
- Connect the follow ripple counter using **T** flip-flops:

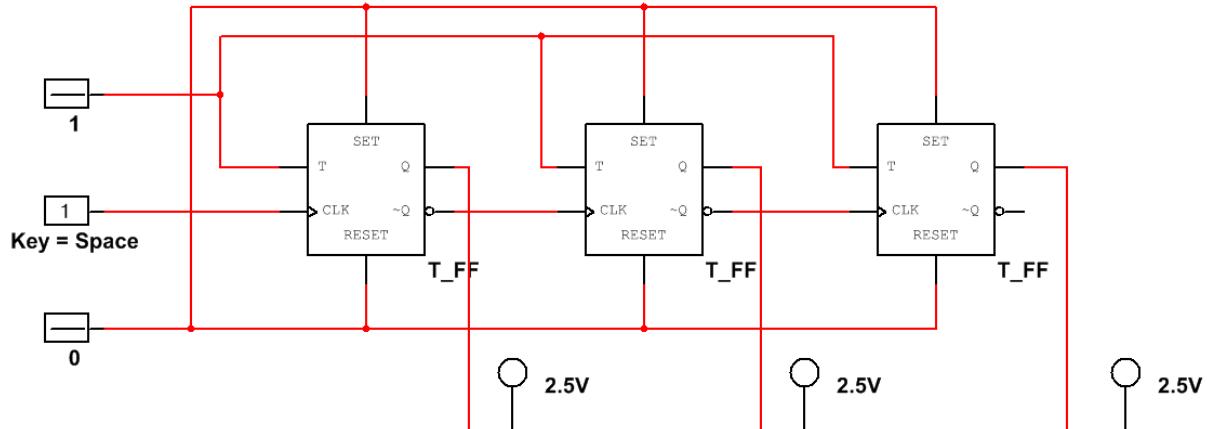


Figure 1-5 Circuit Diagram

- Start he simulation
- Vary the input and observe the probes

1-2 What is the function of the digital constants?

1-3 When does the counter increase its value?

1-4 Describe in your own words how the clock signal cascades through the flip-flops and how that affects the counter.

1-5 How high can this counter count?

- **Stop** the simulation when you are done

1.3 Simulate: Building a Four-Bit Up-Counter

- Launch Multisim
- Connect the following circuit:

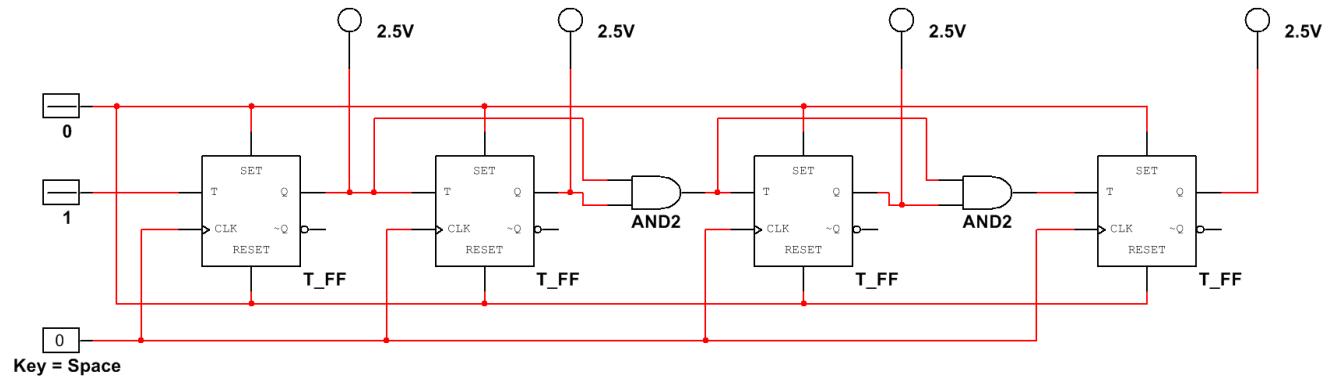


Figure 1-6 Circuit Diagram

- Start the simulation
- Vary the input and observe the probes

1-6 When does the counter increase its value?

1-7 Describe in your own words how the clock signal cascades through the flip-flops and how that affects the counter.

1-8 How high can this counter count?

1-9 How does this circuit compare to the last one you built?

- **Stop** the simulation when you are done

1.4 Simulate: Building a Four-Bit Up-Counter with Enable and Clear

- Launch Multisim
 - Connect the following circuit:

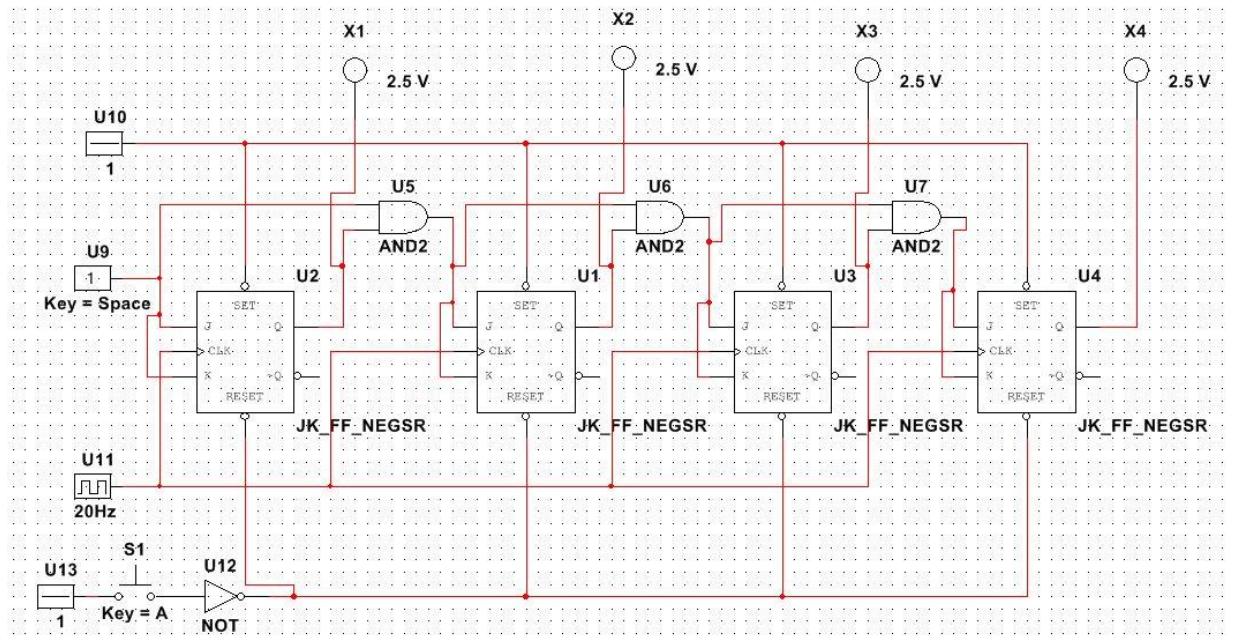


Figure 1-7 Circuit Diagram

The Enable input of the shift register is connected to an interactive digital constant. When high, the counter will increment. The Clear input is connected to an interactive button and is negated because the Clear input of the flip-flops is an active-low line.

- **Start** the simulation
 - Vary the Enable input and observe the probes

1-10 What happens to the counter?

- Click the interactive button or press the **A key** to trigger the Clear and observe the probes

1-11 What happens to the counter?

- **Stop** the simulation when you are done

1.5 Conclusion

1-12 After observing an up-counter (the second circuit), what do you predict would be the behavior of a down counter?

1-13 Do counters have more in common with latches or adders? Explain.

1-14 What determines how high a counter can count? What would you need to do to increase how high it can count?

1-15 In asynchronous counters:

- A. The clock signal of one flip-flop is half a pulse behind the clock signal of the adjacent flip-flop
- B. The clock signal of one flip-flop is half a pulse ahead of the clock signal of the adjacent flip-flop.
- C. The output of one flip-flop is connected to the clock input of the next one.
- D. The clock signal is connected to the clock input of all the flip-flops.

1-16 In synchronous counters:

- A. The clock signals of one flip-flop is in phase with the clock signal of the adjacent flip-flop
- B. The output of one flip-flop is connected to the clock input of the next one.
- C. The clock signal is connected to the clock input of all the flip-flops
- D. None of the above

1-17 What is the biggest drawback in using asynchronous counters?

- A. There is a signal delay between the flip-flops
- B. They don't work at low frequencies
- C. Cannot be built with T flip-flops
- D. All of the above

1-18 What is the role of the Enable input on a counter?

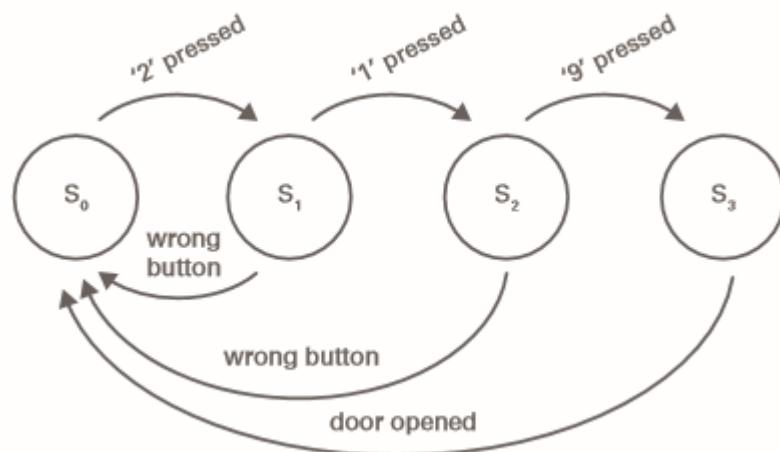
- A. Keep all flip-flops in the circuit at low levels regardless of input
- B. Keep all flip-flops in the circuit at high levels regardless of input
- C. Inhibits counting regardless of clock signal states.
- D. Inhibits counting when the clock signal is at 0

1-19 Synchronous up-counters require which logic gates between flip-flops?

- A. NAND
- B. XNOR
- C. NOT
- D. None of the above

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 12: Finite State Machines

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 12: Finite State Machines

In this lab, you will analyze a new way to model problems: the finite state machine (FSM). An FSM demonstrates the different possible states of a system and the combination of inputs necessary to achieve them. You will explore different types of traffic light systems and how an FSM can be used to model them.

Learning Objectives

In this lab, students will:

1. Gain an awareness of the different methods and components used to create traffic lights.
2. Create a truth table to explore how a circuit behaves in three distinct states.
3. Produce sub circuits to simulate both single and two-way traffic signals.
4. Reinforce understanding of the workings of clock signals and counters.
5. Have the opportunity to use the NI ELVISmx Function Generator

Required Tools and Technology

Platform: NI ELVIS III

Instruments used in this lab:

- Function Generator

Note: The NI ELVIS III Cables and Accessories Kit (purchased separately) is required for using the instruments

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M
- ✓ Install Soft Front Panel support:
<http://www.ni.com/documentation/en/ni-elvis-iii/latest/getting-started/installing-the-soft-front-panel/>

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM_US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Software: NI LabVIEW FPGA Vivado 2014.4

- ✓ Install:
<http://www.ni.com/download/labview-fpga-module-2015-sp1/5920/en/>

Note: Digilent Driver (The installer above automatically downloads the installer below onto your computer)

- ✓ Navigate to:
C:\NIFPGA\programs\Vivado2014_4\data\xicom\cable_driver

s\nt64\digilent
✓ Install: install_digilent.exe

Expected Deliverables

In this lab, you will collect the following deliverables:

- Characteristic table
- Circuit PLD and sub-circuit screenshots
- Two-way traffic light circuit
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

What are Finite State Machines?

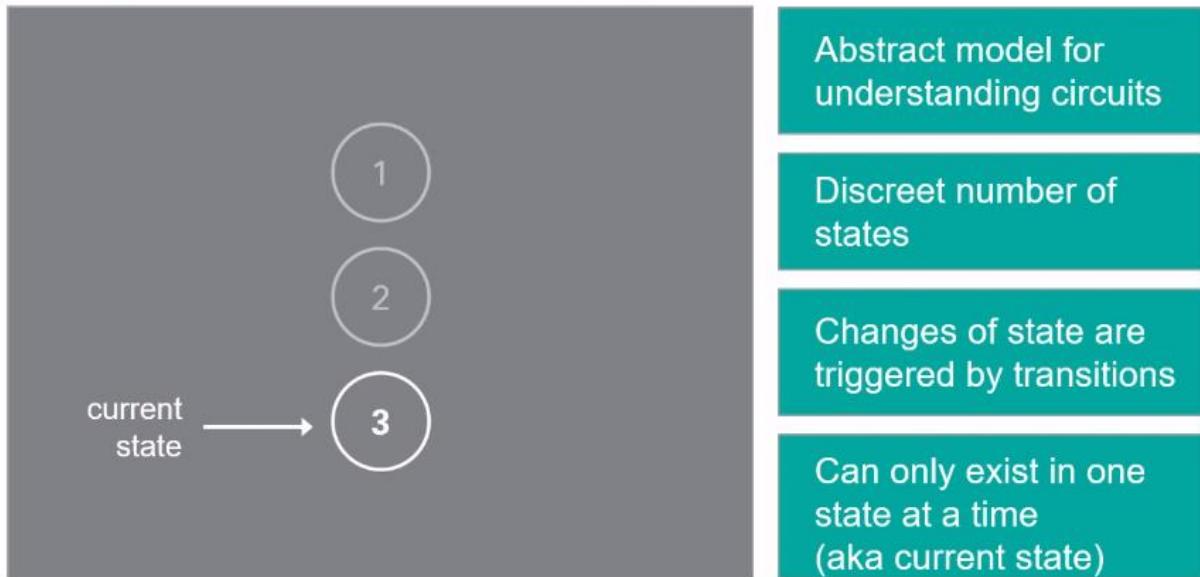


Figure 1-1 Video Screenshot. View the video here: https://youtu.be/I0HBrcE_HOI



Video Summary

- A Finite State Machine is an abstract model for understanding circuits
- There are a discrete number of states that exist in all finite state machines
- Transitions trigger changes of state
- A machine can only exist in one state at any time (the current state)

Finite State Machines

A *finite state machine (FSM)* is an abstract model used to describe the behavior of an object.

- Each circle represents a *state* in the system, while each arrow represents a *transition* to another state.
- Since there are a discrete number of circles, there is only a finite number of states that are possible.
- The machine can only be in one state at any given time. This is known as the *current state*.
- The arrow labels are *events* that trigger the state transition.
- Finite state machines are used in:
 - Vending machines: dispensing a product when the appropriate amount of money is deposited into it.
 - Elevators: dropping people off to the top floors before coming back down.
 - Traffic lights: changing the color sequence at scheduled times.

The example below shows a simple FSM for a system that unlocks when a correct button sequence is pressed. Consider if the example represented a locked door:

- State 0 (**S₀**) represents a state when the lock is enabled
- **S₀** moves to **S₁** after button '2' is pressed.
- **S₁** has two options for the next state – **S₂** if '1' is pressed, otherwise **S₀**.
- **S₃** represents a state where the lock is disabled and enables once the door is opened.

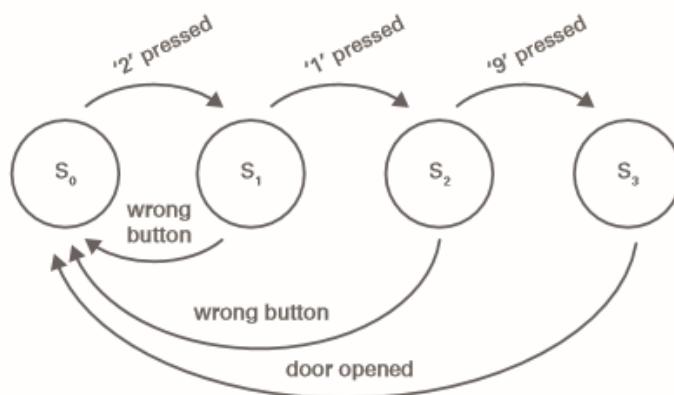


Figure 1-2 Simple Finite State Machine

One way a traffic light signal can be created is by the following components:



Figure 1-3 Traffic light component diagram

1-1 Briefly describe the role of every component in this system and how it relates to the function of the traffic light.

- Create a characteristic table for a state machine with **16 states**, corresponding to **4 inputs**. The **green** light should be on for **8** of the states, the **yellow** light for **3**, and the **red** for **5**.
- For example, in the table below you have four outputs. When all of the outputs are **0**, the **green state is true (1)** and the **yellow and red states are false (0)**.

A	B	C	D	Green	Yellow	Red
0	0	0	0	1	0	0

Figure 1-4 Part of a characteristic table for an FSM with 16 states

Note: Remember, if one of the states is true (1), then the other two must be false (0).

- Attach a drawing, picture, or screenshot of your characteristic table with your completed lab.

1.2 Exercise: Building a Signal Traffic Light in PLD Design

Part 1

- Launch Multisim
- Create a new blank PLD circuit
- Uncheck all connector pins

You are going to create a sub-circuit that uses a clock to increment a 4-bit counter.

- Create a sub-circuit by placing the following components:
 - a 4-bit counter (**CNTR_4BIN_S**)
 - an input connector and name it **Clk**
 - (4) output connectors and name them **QA**, **QB**, **QC**, **QD**
 - an inverter
 - a digital high and a digital low
- Wire the circuit as shown below:

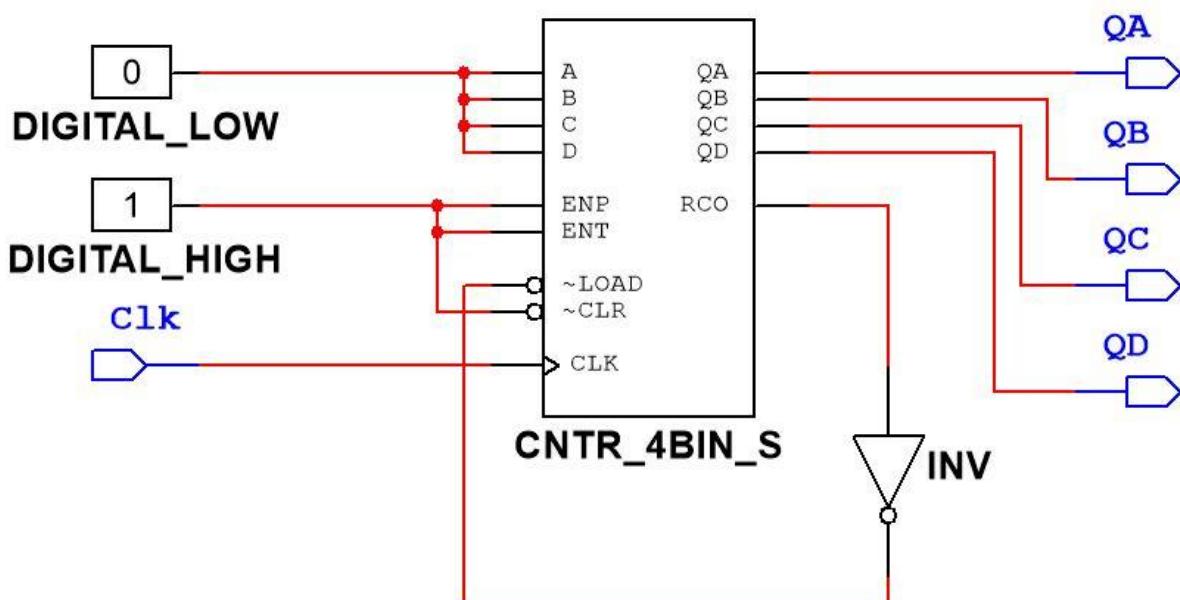


Figure 1-5 Circuit design

Part 2

You will now need to connect the four outputs of the counter (QA, QB, QC, and QD) to a circuit to control the lights.

- Create a new sub-circuit in the main PLD design and label it **Light**
- Open the Light circuit and create three subscripts for the three colors: **Red**, **Green**, and **Yellow**
- Place the following connectors to the sub-circuit:
 - 4 input connectors **QA**, **QB**, , and **QD**
 - 3 output connectors **MainGreen**, **MainYellow**, and **MainRed**
- Use logic gates to create the sub-circuits for individual light colors

Note: You can use the characteristic table you derived previously or, you can complete it now to help define the logic of the three lights.

- Wire the circuit similar to the one below:

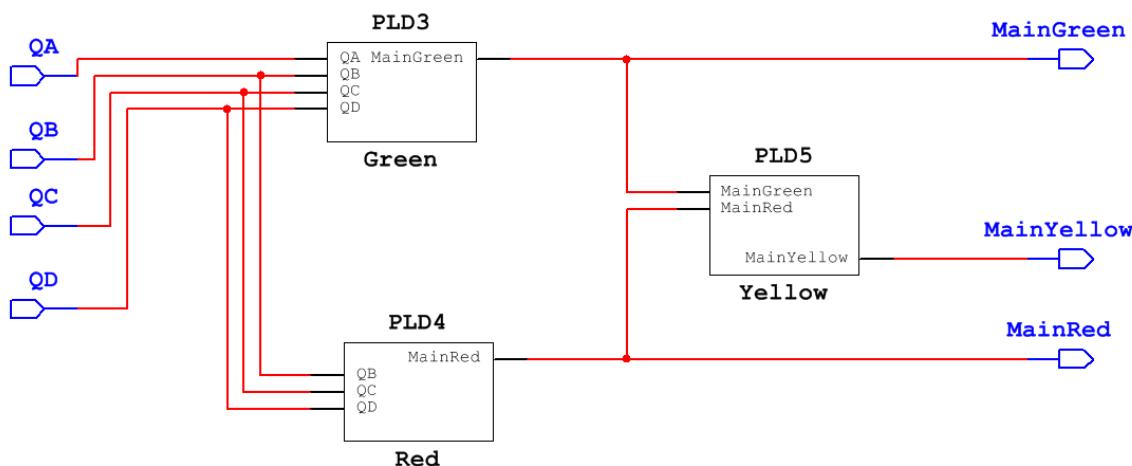


Figure 1-6 Circuit design

Part 3

Now you will test the traffic light in Multisim.

In your main PLD design:

- Connect a digital clock to the **Clk** input of the **Counter** sub-circuit and set it to **5Hz**
- Connect a **red**, **yellow**, and **green** probe to the outputs of the **Light** sub-circuit respectively

When you are done, the circuit should look like this:

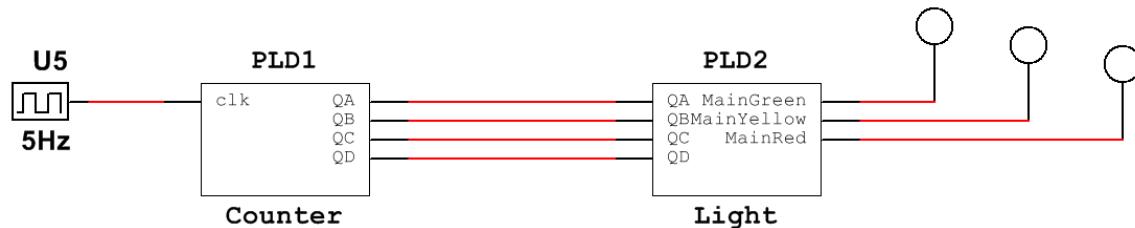


Figure 1-7 Circuit design

1.3 Simulate: Testing a Single Traffic Light in PLD Design

Instructions:

- Simulate the circuit and observe how the lights change over one cycle (green – yellow – red)

1-2 How long does the entire cycle take?

1-3 How long is the green light on?

1-4 Does this correspond to 8 out of the possible 16 states?

- **Stop** the simulation when you are done

1.4 Implement: Using a Function Generator

Setting Up the Function Generator

Next, you need to deploy the design to the Digital Electronics Board. However, the digital clock component on Multisim cannot be deployed. In order to create a clock pulse for the counter of your traffic light, a Function Generator has to be used. The NI ELVISmx Function Generator is a program that allows you to customize frequency waveforms sent to the FPGA via a BNC breakout plug connected to the ELVIS platform.

- Connect the BNC plug to the FGEN port of the ELVIS platform. Connect the red clip to a wire and plug the wire into the output component **BB_S_DIO4**. Similarly, connect the black clip to a wire and plug it into the **GND**.
- Make the following changes to the default setting of the Function Generator:
 - Waveform: **Square**
 - Frequency: **5 Hz**
 - Amplitude: **3.00 Vpp**
- Make sure that the Device (**NI ELVIS III**) and Signal Route (**FGEN BNC**) are successfully recognized by the Function Generator. In order to be recognized, the Elvis board must be turned on.

Use the following image as a reference for the settings:

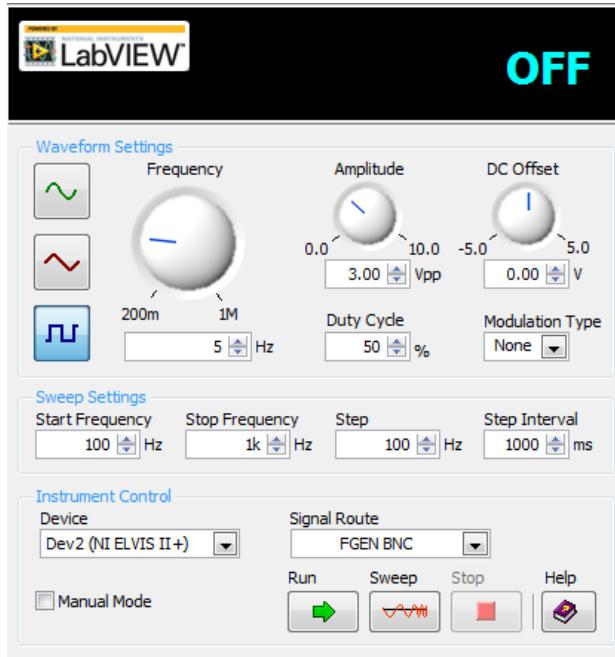


Figure 1-8 Settings reference

Deploying to the Digital Electronics Board

To deploy the design to the Digital Electronics Board, the digital clock and probes need to be replaced with PLD components.

- In the main PLD design, replace the probes with output connectors **BB_S_DIO0** and **BB_S_DIO2**
- Replace the digital clock with **BB_S_DIO4** (which is connected to the Function Generator)

Your main PLD design should look like this:

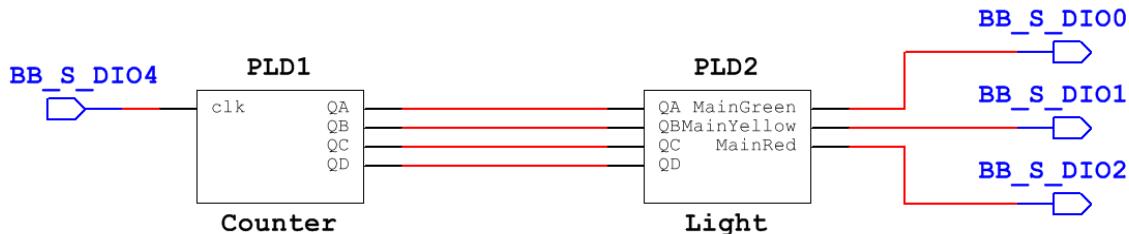


Figure 1-9 PLD design

- On the Digital Electronics Board, wire a **green**, **yellow**, and **red LED** to the corresponding outputs.
- To wire a LED to the board, connect a **1000 Ω resistor** to the long tip of the LED and plug it into the corresponding output connector on the bread board.
- Plug the short tip of the LED into the **GND** output connector on the breadboard.
- Deploy the circuit to the Digital Electronics Board.
- Create a sketch, take a picture, or include a screenshot of your finished circuit PLD, and the individual sub-circuits with your completed lab.

1.5 Exercise: Two Way Traffic Lights

Sometimes, traffic lights have a fourth light that allows drivers to turn left while opposing traffic is stopped. This light is called an advanced green light and is usually on for a short period of time while the red light is also on.

- For example, when the light of the main circuit is green, the light of the new circuit should be red.
- Take a screenshot, draw a picture, or take a photo of the new circuit and include it with your finished lab.

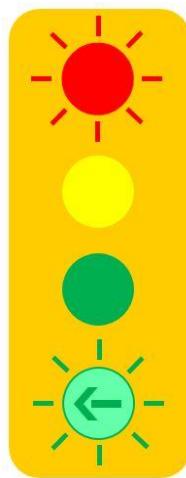


Figure 1-10 Advanced green traffic light

Using the circuit for the first traffic light as a template, add a fourth state for advance green traffic light.

Note: You will need to consider how long you would like it to appear for (truth table) and you will need to add another light color to your circuit with logic gates.

- Create a sketch, take a picture, or include a screenshot of your new circuit with your completed lab.

1.6 Conclusion

1-5 In your own words, describe how the clock controls the three traffic lights.

1-6 What is the benefit of breaking down the circuit into PLDs?

1-7 What would be required to create more total states?

1-8 Which of the following describes the behavior of a finite state machine (FSM)?

- A. There are a finite number of states
- B. The machine can be in a maximum of two states at a time.
- C. A change of state can occur at any time spontaneously
- D. All of the above

1-9 FSMs are used in:

- A. Vending machines
- B. Coffee machines
- C. Televisions
- D. Car brakes

1-10 A traffic light signal is created with which of the following components?

- A. Gated SR latch, Counter, LED
- B. Timer, Counter, LED
- C. Timer, BCD to binary converter, LED
- D. Timer, D flip-flop, LED

1-11 In the truth table of a traffic light system, why is only one color allowed to be true
(1) at any given time?

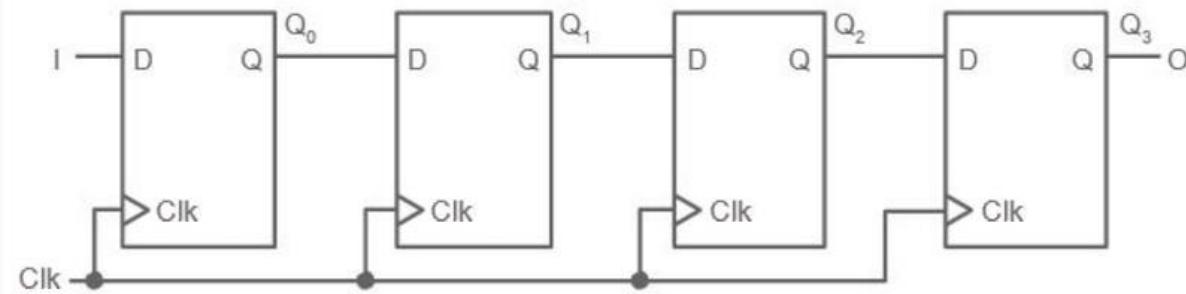
- A. There is only one LED that gates reset each time the color changes
- B. Attempting to light up two colors simultaneously would short the circuit
- C. FSMs can only exist in one state at any given time
- D. All of the above

1-12 The input into the traffic light state machine PLD:

- A. An interactive digital constant
- B. Digital constant
- C. Digital clock
- D. None of the above

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 13: Shift Registers

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 13: Shift Registers

In *Lab 11: Counters* you explored how flip-flops can be implemented to create counters. In this lab, you will explore how flip-flops can be arranged to create shift registers. A shift register is a group of flip-flops arranged so that the stored bits are shifted either left or right, effectively multiplying or dividing the input respectively.

Learning Objectives

In this lab, students will:

1. Observe the basic function of a four-bit shift register using D flip-flops
2. Examine how changing the clock frequency of the input of a shift register changes the time taken for the initial signal to reach the flip-flops
3. Explore waveforms produced in serial serial-in/parallel-out shift registers to explain how the clock signal travels across flip-flops

Required Tools and Technology

Platform: NI ELVIS III

Instruments used in this lab:

- Function Generator
- Logic Analyzer

Note: The NI ELVIS III Cables and Accessories Kit (purchased separately) is required for using the instruments

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M
- ✓ Install Soft Front Panel support:
<http://www.ni.com/documentation/en/ni-elvis-iii/latest/getting-started/installing-the-soft-front-panel/>

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
 - ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>
-

Expected Deliverables

In this lab, you will collect the following deliverables:

- Timing diagram
- Screenshot (or similar) of the waveforms produced
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

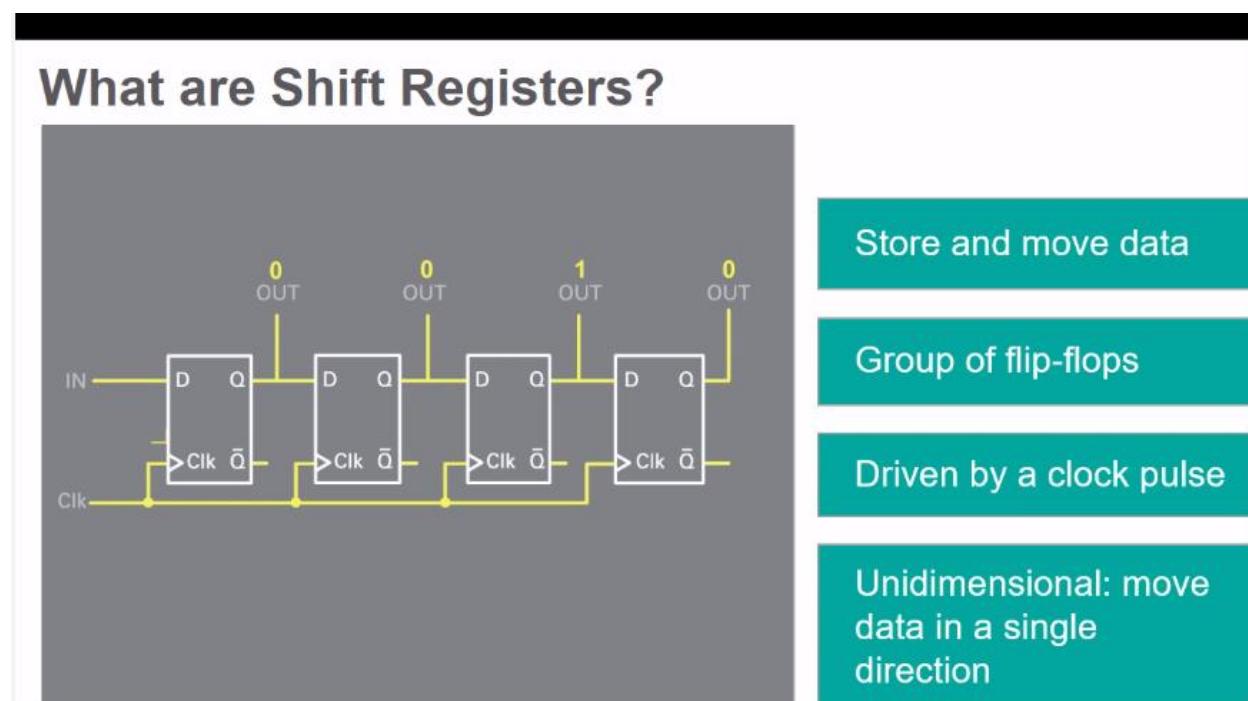


Figure 1-1 Video Screenshot. View the video here: <https://youtu.be/59WfRFp54Zs>



Video Summary

- Shift registers store and move data
- Shift registers are driven by a clock pulse
- Shift registers that move data in a single direction are called unidirectional and shift registers that can reverse direction are called bidirectional

Shift Registers

Shift registers are circuits that store and move data, made of a group of flip-flops. The group of flip-flops is arranged so that the stored data is shifted from one to a neighboring one for every clock pulse in a selected direction. They are used in serial/parallel conversion, serial data transfer, arithmetic functions and delay elements.

The binary multiplication of a number by 2 is performed by shifting its bits to the left and inserting a 0 on the least significant bit position. By shifting all the bits of a given binary number one position to the right, it is divided by 2.

The figure below presents the simplest shift register, which is created by connecting the output of each flip-flop to the input of the flip-flop to its right.

- It shifts its contents one position to the right, being a unidimensional shift register.
- The data is loaded into the shift register serially through the input **I** and the contents of each one of the flip-flops is transferred to the next one (for the last one to the **O** output) on the rising edge of the clock signal.
- **I** is a serial input, while **O** is the serial output of the shift register.

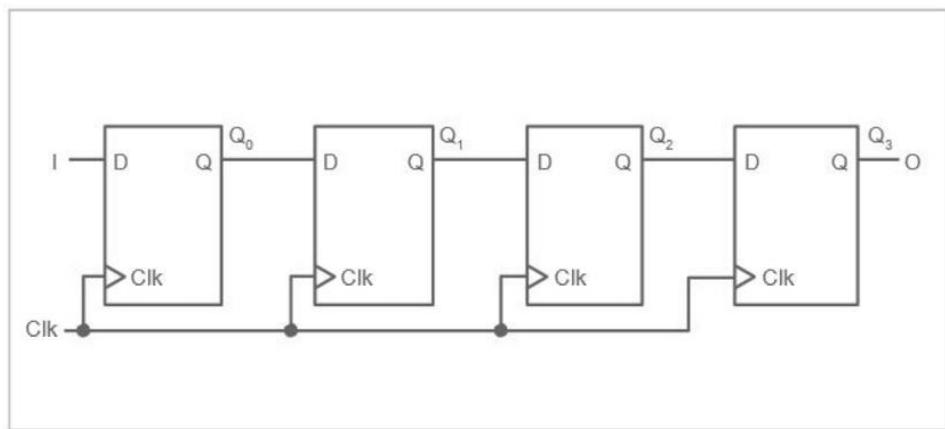


Figure 1-2 Simplest shift register

Types of Shift Registers

There are four types of shift registers:

1. *Serial-in/Parallel-out (SIPO)* where serial data is loaded into the register one bit at a time and available at the output in parallel simultaneously.
2. *Serial-in/Serial-out (SISO)* where the data enters and exists the shift register one bit at a time
3. *Parallel-in / Serial-out (PISO)* where the data is loaded simultaneously but is shifted out one bit at a time
4. *Parallel-in/Serial-out (PIPO)* where the data is loaded at the same time and shifted together

Shift registers can only be implemented using edge-triggered circuits and not level-sensitive gated latches. When using level sensitive circuits, the value of the input can propagate through more than one of them during the period in which the clock signal is 1.

The figure below shows the way in which the data stored by the shift register changes depending on the I input, during a sequence of eight clock cycles and assuming that the initial state of all flip-flops is 0.

	I	Q ₀	Q ₁	Q ₂	Q ₃ = 0
t ₀	1	0	0	0	0
t ₁	0	1	0	0	0
t ₂	0	0	1	0	0
t ₃	1	0	0	1	0
t ₄	0	1	0	0	1
t ₅	0	0	1	0	0
t ₆	0	0	0	1	0
t ₇	1	0	0	0	1

Figure 1-3 Shift register truth table

The figure below shows the graphical symbol of a four-bit shift register:

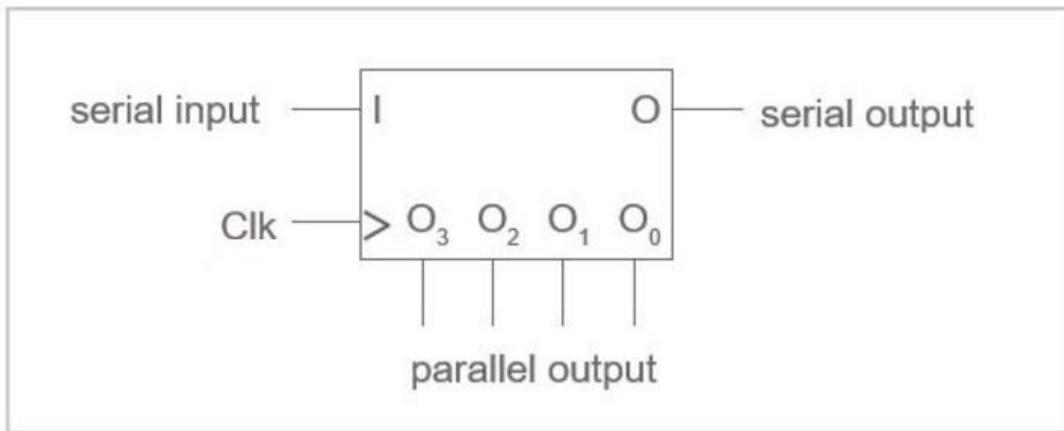


Figure 1-4 Shift register graphical symbol

Shift registers can also be built using J K flip-flops, as shown in the figure below:

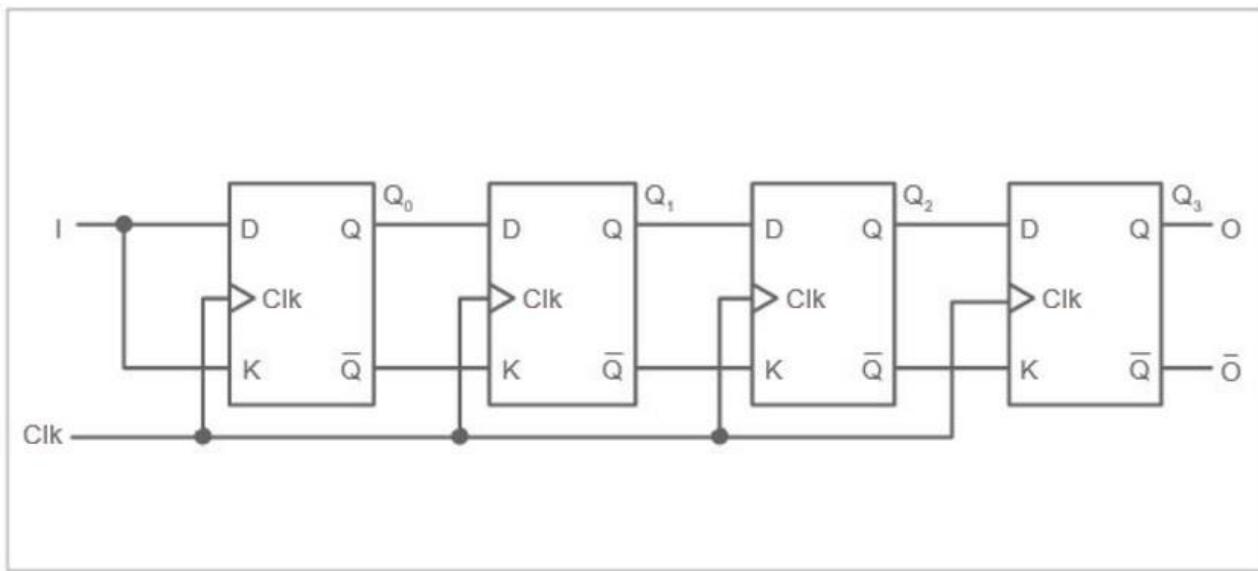


Figure 1-5 Shift register built out of JK flip-flops

1-1 What would be the main differences in structure and function between unidirectional and bidirectional (reversible) shift registers?

- Sketch a timing diagram using the table from above. Show how the clock signal travels along the flip-flops. Include this sketch with your completed lab.

1.2 Exercise: Building and Testing a Four-Bit Shift Register

Instructions:

- Launch Multisim
- Connect the following circuit:

Note: The digital constant needs to be set a **0** and the clock to **10 Hz**

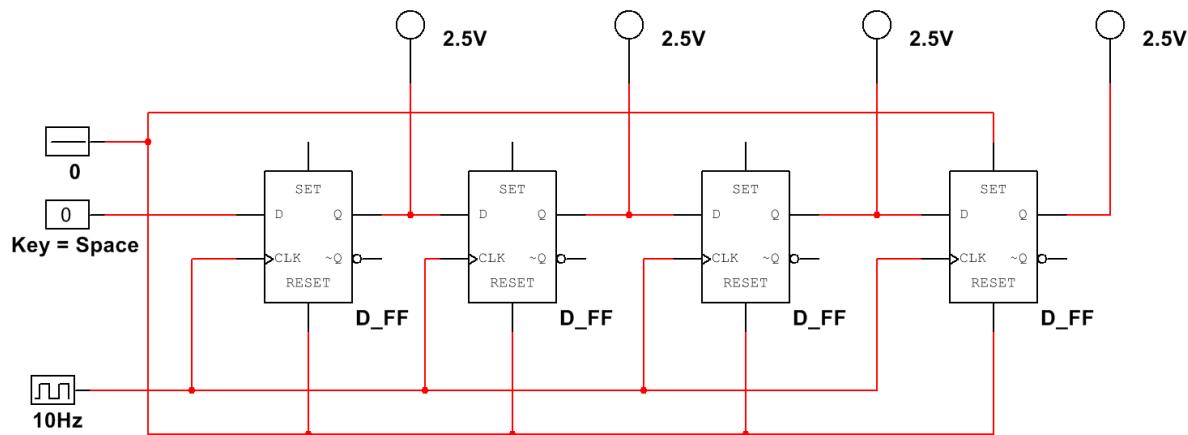


Figure 1-6 Circuit diagram

- **Start** the simulation
- Vary the interactive digital constant from **0** to **1**

1-2 What happens to the probes?

- Reset the circuit

1-3 What is the time taken between lighting of the first and second probes?

- **Stop** the simulation when you are done
- Change the frequency of the clock input to **20 Hz** and run the simulation again

1-4 What is the time taken between the lighting of the first and second probes?

- **Stop** the simulation when you are done
- Change the frequency of the clock input to **5 Hz** and run the simulation again

1-5 What is the time taken between the lighting of the first and second probes?

- **Stop** the simulation when you are done

1-6 Describe and explain the pattern observed of the speed at which the first and second probes light up as the clock frequency increases.

1-7 What kind of shift register is this?

1.3 Exercise: Building and Testing a Serial-In/Parallel-Out Shift Register

Instructions:

- Create a new **Blank** design in Multisim
- Build the following circuit using **D Flip Flops**, a **Logic Analyzer (XLA1)** and two **Function Generators (XFG1 and 2)**:

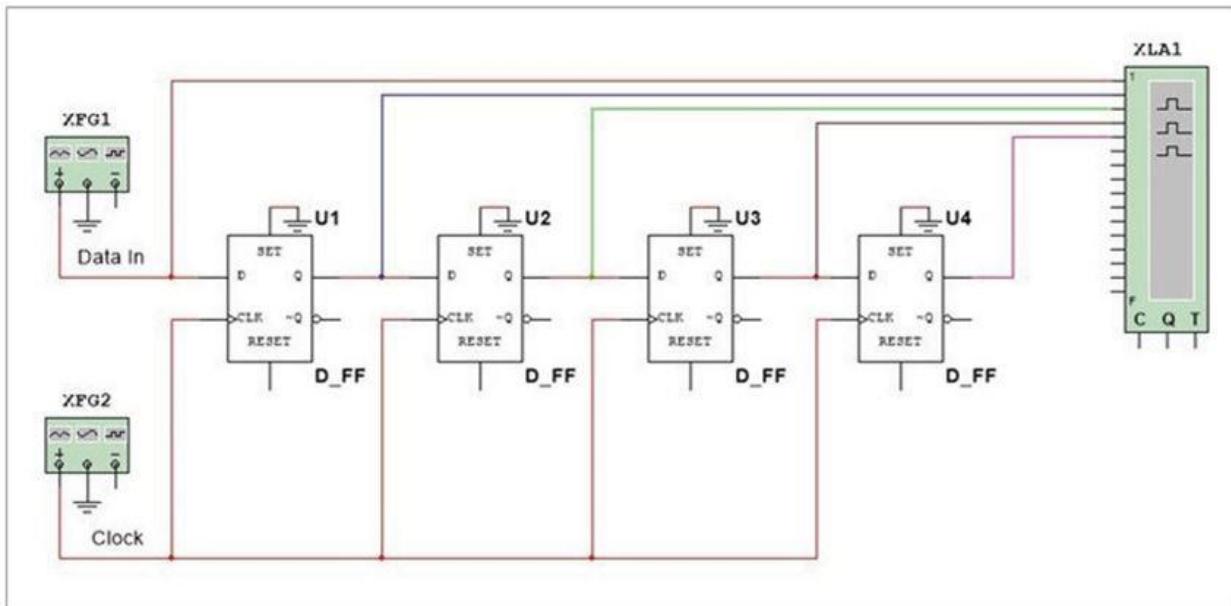


Figure 1-7 Circuit diagram

- Double-click on the function generator for the **clock input**. Configure it with the settings shown:

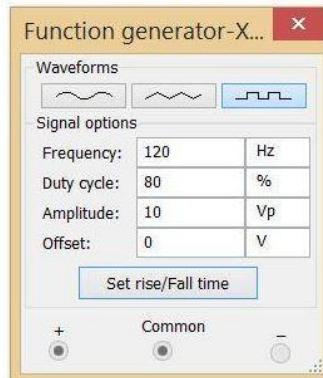


Figure 1-8 Function generator clock input settings

- Double-click on the function generator for the **data input**. Configure it with the settings shown:

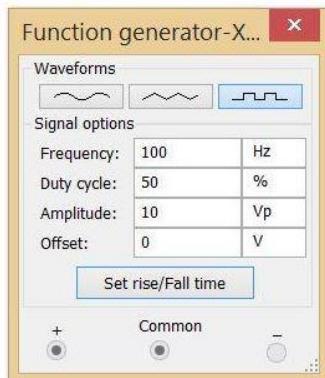


Figure 1-9 Function generator data input settings

- Double-click on the **Logic Analyzer** to open its front panel
- Run the simulation and note the output waveforms (timing diagrams)
- **Stop** the simulation after it has completed one full screen on the logic analyzer
- Take a picture, draw a sketch, or take a screenshot of the waveforms produced and include it with your finished lab

1-8 Explain what the waveforms above are telling you about the transmission of a clock signal.

1.4 Conclusion

1-9 Counters and shift registers can both be created using D and JK flip-flops. Why is it possible to create a counter from a T flip-flop but not a shift register?

1-10 Recently we have been able to create a new class of shift registers that are called universal. Using your knowledge of the four different types of shift registers, hypothesize what would be its function(s).

1-11 Shift registers can only be configured to shift data in one direction.

- A. True
- B. False

1-12 Which of the following is an example of a shift register?

- A. Serial-in/Parallel-in
- B. Parallel-in/Serial-in
- C. Serial-out/Parallel-out
- D. Parallel-in/Serial-out

1-13 What components are shift registers made of?

- A. Adders
- B. SR Latches
- C. Flip-flops
- D. None of the above

1-14 What happens when the clock frequency is changed at the input?

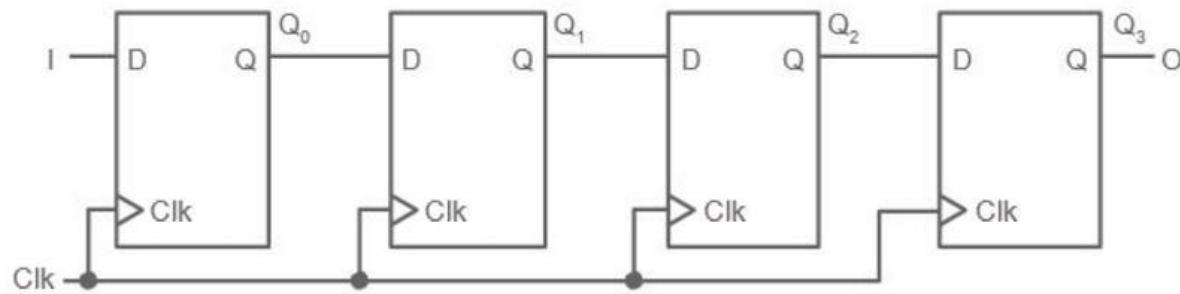
- A. Changes which direction data shift in
- B. Changes the time taken for the data to be shifted between flip-flops
- C. Changing the clock frequency has no effect
- D. Keeps the shift register from working properly

1-15 Can shift registers be implemented with level-sensitive gates latches?

- A. Yes
- B. No

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 14: Semiconductor Memory

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 14: Semiconductor Memory

Memory is a type of circuit used to store electronic data. Memory cells are tiny circuits existing in a larger chip that store one bit of binary data each. We have already seen examples of circuits that can be used in memory cells, such as various flip-flops. In modern electronics, transistors are used in most memory chips.

Memory chips can usually *read* and *write*. Writing is a process where data is stored in groups of memory cells called words. Words consist of bits to the power of 2 (i.e. 2, 4, 8, 16 bits etc). Each word has a memory address that acts as a marker so that they can be identified when they need to be read. When a chip reads, it recalls the data stored in a word.

Learning Objectives

In this lab, students will:

1. Configure a word generator in Multisim
2. Observe the reading and writing of a 2-bit code on a 2K8 RAM chip
3. Design, construct and simulate the writing and reading of a 4-bit code on a 2K8 RAM chip

Required Tools and Technology

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Expected Deliverables

In this lab, you will collect the following deliverables:

- Screenshot of RAM cell
- Circuit screenshot
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

Semiconductor Memory

Introduction Video



Figure 1-1 Video Screenshot. View the video here: https://youtu.be/V0yzDRDU_BU



Video Summary

- Memory is a type of circuit used to store electronic data
- Memory cells are tiny circuits contained in a larger chip that store 1-bit of binary data in each
- Technology and methods are improving and reducing the amount of time needed to read data

RAM and ROM

There are increasing technologies and methods that exist to reduce the time needed to read data. For example, the memory in CDs can only be read in the order it was written. Even though this process takes a matter of nanoseconds, others are even more efficient. Some of the newer technologies include *RAM (Random Access Memory)*.

- Random Access Memory can be both written, read and the memory accessed in any order.
- RAM is commonly found in computers.
- It is only accessible when the device is turned on. When turned off, none of this information can be accessed.

Some forms of semiconductor memory can only be read and not written. These are known as *Read Only Memory (ROM)*. Some examples of ROMs are:

- *PROMs (Programmable Read Only Memory)* : The data can be written on the chip only once
- *EPROMs (Electrically Erasable Programmable Read Only Memory)*: Data written on the chip can be erased with UV light allowing the chip to be rewritten with data.

RAM Cell

Instructions:

- Using external resources (i.e. internet, books), draw a sketch of a RAM cell using logic gates and an SR flip-flop.

Your diagram should include the following labels:

- Input
- Select
- Output
- Read/Write

Note: Draw a sketch, take a picture, or take a screenshot of your RAM cell and include it with your finished lab.

1-1 What are the differences between RAM and ROM?

1.2 Exercise: Writing and Reading of 2-Bits of Data to Memory

Part 1

- Launch Multisim
 - Open a new **Blank** circuit design
 - From the right-hand panel, place a word generator on your circuit (**XWG**)
 - Double-click on the **word generator** and configure it as the one shown:

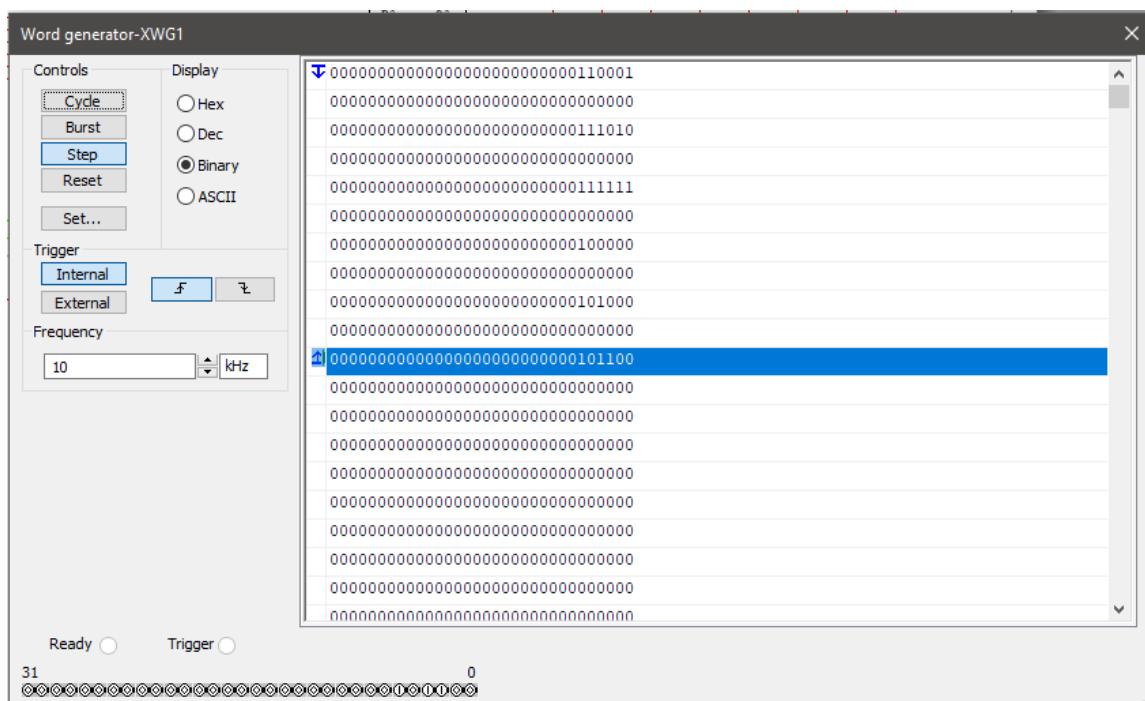


Figure 1-2 Word generator configuration

- The first line of binary code should have a blue arrow pointing down (see *Figure 1-2*). This indicates to the word generator to use this line as the starting point.
 - If not, right-click this line and select **Set Initial Position**.
 - Right-click on the **eleventh line** of binary code and select **Set Final Position**. All of the lines below this are filled with zeros, so you don't need to do anything with these.

Part 2

- Connect the rest of the circuit as shown below:

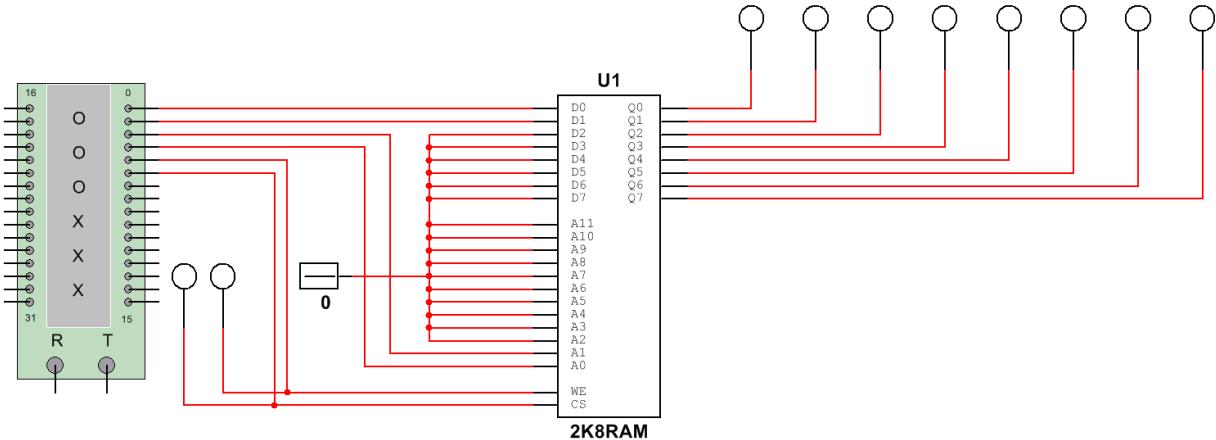


Figure 1-3 Circuit diagram

The pinout for the 2K8 RAM is:

- D0-D7 - Digital inputs
- A0-A11 - Addresses
- WE - Write when high and read when low
- CS - Address select
- Q0-Q7 - Digital outputs

In this example, two bits of data will be written to **D0** and **D1** and addresses **A0** and **A1** are used, the remaining digital inputs and address pins are set to zero. We will be writing data and then be reading the data we have written as indicated in the table. Note that each command is followed by a line full of zeros. The line of zeros simulates the falling edge of a clock so that the next command can run on the rising edge of a clock.

CS	WE	A0	A1	D0	D1	Description
1	1	0	0	0	1	Write 01 to address 00
0	0	0	0	0	0	Next clock cycle
1	1	1	0	1	0	Write 10 to address 01
0	0	0	0	0	0	Next clock cycle
1	1	1	1	1	1	Write 11 to address 11
0	0	0	0	0	0	Next clock cycle
1	0	0	0	0	0	Read data from address 00
0	0	0	0	0	0	Next clock cycle
1	0	1	0	0	0	Read data from address 01
0	0	0	0	0	0	Next clock cycle
1	0	1	1	0	0	Read data from address 11

Figure 1-4 Command table

Part 3

Instructions:

- Double-click the **Word Generator** to expand the interface
 - Right-click on the first row of binary data and select **Set Cursor** in the context menu.
 - Click the **Step**.
 - Select **Simulate >> Run** or click the **Run** icon.
- Click the **Step** button to step through the data, wait for the probes connecting to WE and CS pin to change state before you click **Step** again.
- Repeat until you have reached the final input.

1-2 Which probes lit up when address 11 was read?

1-3 What would happen if you moved the final input to line 13?

1-4 Most of the inputs on this chip are wired to zeros. If you wanted to use those lines, what would they allow you to do that this sample circuit couldn't do?

1.3 Simulate: The Reading and Writing of 4-Bits of Data

Instructions:

- Use the circuit and table created in Part 1 and expand on it to design and build a circuit which writes and reads 4 bits of data instead of 2.

Note: you will need to make significant changes to the circuit in its current form.

- Modify the Word Generator to test that your circuit stores and recalls the data in the correct memory addresses.
- Draw a sketch, take a picture, or take a screenshot of your new circuit and include it with your finished lab.

1-5 What modifications did you have to make?

1.4 Conclusion

1-6 Could we have used an EPROM chip in your experiment instead of a RAM? Explain.

1-7 When manufacturers are considering which chips to put into their products, what are some of the factors they would consider?

1-8 In modern electronics, what is the most common element in memory chips?

- A. Transistors
- B. Resistors
- C. Flip-flops
- D. Counters

1-9 Semiconductor memory can be:

- A. Read and written
- B. Read only
- C. Written only
- D. None of the above

1-10 The main drawback of RAM memory is:

- A. It can only be read
- B. It is not very different
- C. Cannot store large amounts of data
- D. Is inaccessible when device is turned off

1-11 Each horizontal row on a word generator represents:

- A. 2 bits of stored data
- B. One word
- C. The rising clock cycle
- D. None of the above

1-12 What is the purpose of the WE input on the 2K8 RAM chip?

- A. Address select
- B. Digital input
- C. Write when low and read when high
- D. Write when high and read when low

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 15: Digital Dice

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 15: Digital Dice

In this lab, you will be using the various different concepts and skills acquired in the Digital Electronics Lab series to construct a digital dice that will generate a random number when the circuit is turned on.

You will be provided with the sequence of tasks and some resources that will guide you through the process. This will need to be supplemented with some individual research in order to complete the task.

Some resources that may be valuable are:

- Previously completed labs
- Peer collaboration
- NI.com website

Learning Objectives

In this lab, students will:

1. Use a sequence of steps grounded in prior knowledge of digital circuits to create a digital dice that generates random numbers when activated
2. Strengthen research and problem-solving skills as they conceptualize and build the digital dice circuit one step at a time

Required Tools and Technology

Platform: NI ELVIS III

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>

Hardware: Electronic Components

Components used in this lab:

- (7) LEDs
- (7) 1000 Ω resistors
- (7) wires

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Software: NI LabVIEW FPGA Vivado 2014.4

- ✓ Install:
<http://www.ni.com/download/labview-fpga-module-2015-sp1/5920/en/>

Note: Digilent Driver (The installer above automatically downloads the installer below onto your computer)

- ✓ Navigate to:
C:\NIFPGA\programs\Vivado2014_4\data\xicom\cable_driver\s\nt64\digilent

- ✓ Install: install_digilent.exe

Expected Deliverables

In this lab, you will collect the following deliverables:

- Dice pattern truth table
- Sketched and actual logic circuits for the dice
- Observation notes
- [Optional] final circuit
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

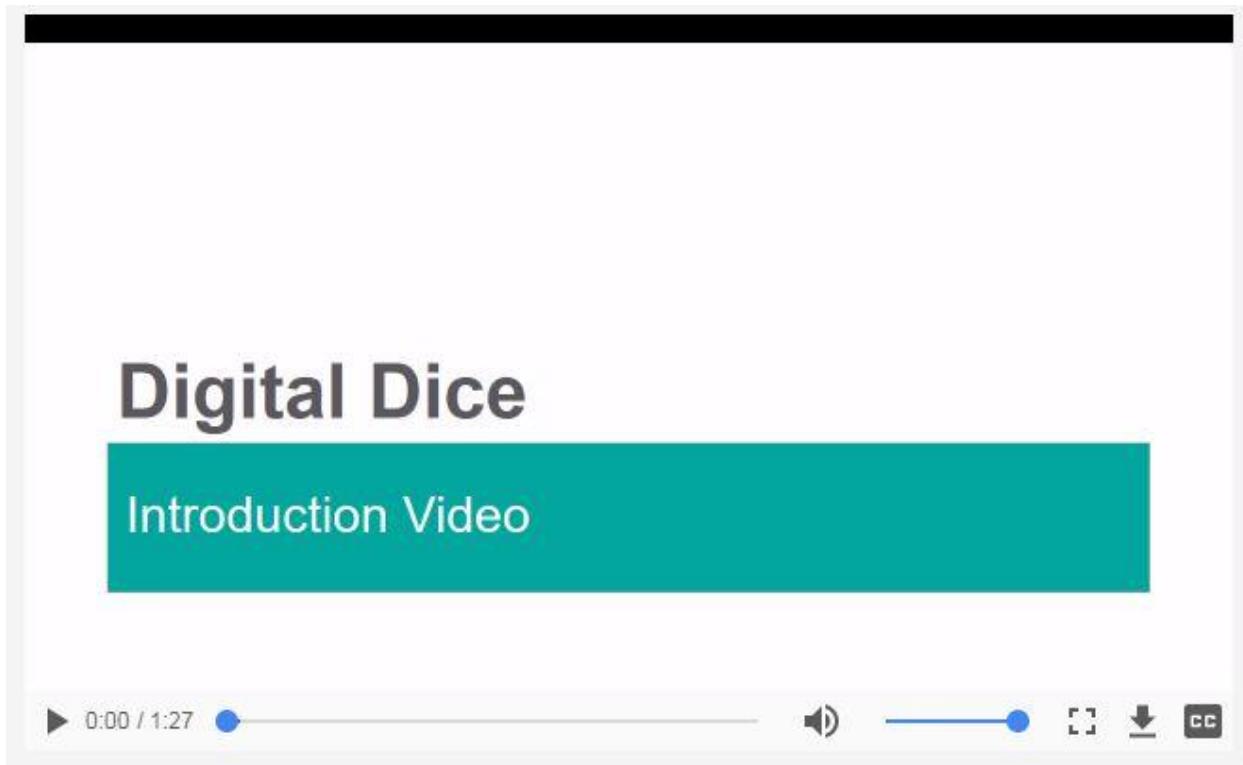


Figure 1-1 Video Screenshot. View the video here: https://youtu.be/acDW6k_ox1w



Video Summary

- This video explores how you will tie together what you have learned in this course to design and build digital dice
- Digital dice are made using a circuit that generates a random number every time it is turned on
- You will use truth tables, combinational logic circuits, BCD to 7-segment display decoders, counters, clock pulses and encoders

PLDs, FPGAs, and VHDL Code

An FPGA (field-programmable gate array) is one type of PLD (programmable logic device). Other PLDs exist such as generic array logic devices (GALs) and complex programmable logic devices (CPLDs).

- Something they all have in common is a need to store and communicate their configuration.
- To store the data of their configuration, PLDs often use:
 - SRAM
 - Flash memory
 - EPROM
- To communicate their configurations, there are many programming languages, and chip manufacturers often have their own tools which can convert designs into a configuration.
- The Digital Electronics Board uses Xilinx tools to convert circuit designs into something that the FPGA can use. Xilinx tools can also be used to generate VHDL code from a circuit design.
- VHDL is a programming language commonly used across many chip types, PLD types, and manufacturers. It is one of the most standard and standardized languages used for designing PLDs.

Dice Patterns

Consider the case of a single die. On each of its six sides, one of the following patterns appears, representing the numbers one through six.

These patterns are traditional:



Figure 1-2 Traditional die faces

You can also think of them as seven lights arranged in an "H" pattern, as shown below.

Note: This pattern allows you to create any one of the six configurations on the face of a die.



Figure 1-3 "H" pattern

On closer inspection, there are only four unique patterns from which the pattern for any face can be formed.

- These four patterns, shown below, are called base patterns **A**, **B**, **C**, and **D**.
- The base state A has only one light and it occupies the center position.
- States B, C, and D all use two lights occupying one of the two diagonal positions or the horizontal position.

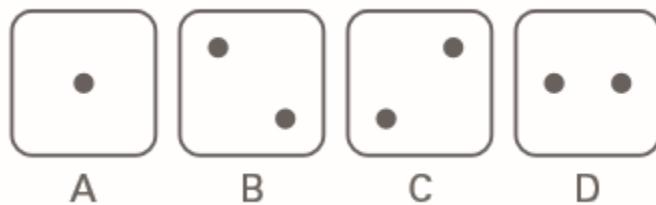


Figure 1-4 Four base states

1-1 Using the information from the previous step, complete the following truth table for the absence or presence of the base patterns.

Die Face	A	B	C	D
1				
2				
3				
4				
5				
6				

1-2 Use the truth table from *Question 1-1* to sketch out the different logic circuits for each of the six sides of the dice. Think about the different ways you can input the numbers 1 through 6 into these logic circuits.

Note: Take a picture, draw a sketch, or take a screenshot of your circuits and include them with your finished lab.

1.2 Implement: Testing Circuits in Multisim

Instructions:

- On your computer, launch Multisim.
- Place and connect the circuits you sketched in the previous part.
- Use probes to simulate the lights in the "H" pattern.
- **Start** the simulation and test out your circuits.
- If your circuits are not working properly, review the previous steps.
- If your circuits are working properly, continue to the next step.

Note: Take a picture, draw a sketch, or take a screenshot of your circuits and include it with your finished lab.

1.3 Simulate: BCD to 7-Segment Display Decoder

A Seven Segment Display (SSD), as shown in *Figure 1-5*, consists of individual segments that can be illuminated to display any one of the numbers from 0 to 9.

For example, number 7 can be created using segments **a**, **b**, and **c**.

If you wanted to use the concept of illuminating parts of this pattern (similar to the die), you could do so using a BCD to 7-Segment Display Decoder.

In Multisim, there are two groups of chips that can perform this function:

- 74LS47
- 74LS48

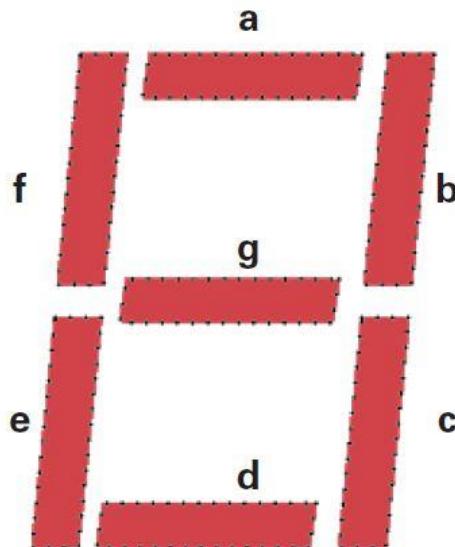


Figure 1-5 7-Segment display (SSD)

1-3 Observe both components in Multisim. What are their similarities and differences?

- Go online and discover more about these products using datasheets.

1-4 What can you add to your list of similarities and differences above?

1-5 If you had to choose one of these components to use in an experiment using BCD to SSD conversion, which one would you choose and why?

- Launch Multisim
- Connect the chosen BCD to 7-Segment Display Decoder to both **interactive digital constants** (inputs) and an **SSD** (output)
- **Run** the simulation
- If the chip does not perform the desired function, **stop** the simulation and replace with the other one

1.4 Implement: Modulo-6 Counter

A *modulo-6 counter* is any counter with six unique states that repeat the sequence every six counts. It can be a binary counter with decimal equivalents of (1, 2, 3, 4, 5, 6, 1...) or it can be a random number generator that may produce an output such as (5, 1, 6, 4, 3, 2, 5...). In this lab, you will be building a simpler modulo-6 counter using three D flip-flops configured as a switched tail ring counter. An example modulo-6 counter CLC is shown below, along with the corresponding characteristic table. Note that the inverted output of the last element is connected back to the input of the first element. On command from a rising edge of the clock pulse, all the D inputs shift to their respective outputs. Each clock pulse generates a new sequence of outputs on Q1, Q2, and Q3, which repeats after six pulses.

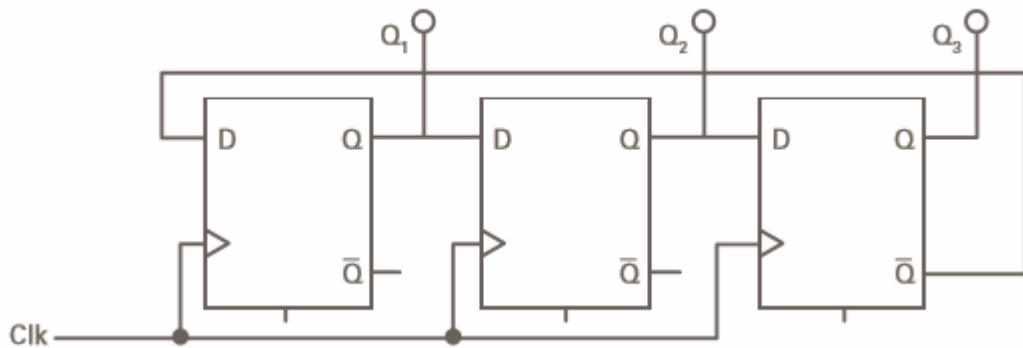


Figure 1-6 Modulo-6 counter combinational logic circuit

Cycle	Q ₁	Q ₂	Q ₃	
1	0	0	0	
2	1	0	0	
3	1	1	0	
4	1	1	1	
5	0	1	1	
6	0	0	1	
7	0	0	0	same as cycle 1

Figure 1-7 Modulo-6 counter characteristic table

Instructions:

- Launch Multisim.
- Build the modulo-6 with switched tail ring counter.
 - Probes should be placed at the outputs **Q1**, **Q2**, and **Q3**.
- **Start** the simulation and observe how the probes light up.

1-6 Is there a pattern or is it random?

- **Stop** the simulation

1-7 How does a switch tail counter differ from the counters you created in the *Counters* lab?

1-8 Go online and find out more about modulo-n counters. Explain how they are named.

1-9 Is this modulo-6 counter a synchronous or asynchronous counter? Describe.

1-10 Could a ripple counter be used instead of the switched tail ring counter to perform the same behavior?

1.5 Exercise: Building the System Clock

To roll the dice, you can use a high clock speed. A clock frequency of several kilohertz is fast enough so that the eye cannot tell which numbers are appearing on the probes. It looks just like the dice are rolling.

- Launch Multisim
- Place a clock signal with a **5 kHz** frequency
- To stop the dice rolling, you need only to stop the clock. You can simulate this feature by simply stopping the simulation

1.6 Exercise: Building the Three to Four Line Encoder

The encoder takes the three counter outputs, Q_1 , Q_2 , Q_3 , and sets the base states, A , B , C , and D , to represent the dice numbers 1 through 6.

Note: Previously you had no reason to decide which output corresponded with which count. Now, you will need to make those choices. If you utilize foresight, this will make your choices easier.

In the table below are the possible states for a modulo-6 switch tail ring counter. You will notice that for each of the states Q_1 , Q_2 , Q_3 there is a prime at the right of the table. This means that the output of the ring counter is fed back into the input. This Q' are opposite states of their original. For example, when Q_1 is 0, Q'_1 is 1. Each output has three (1) states and three (0) states. You could use one of these outputs, say Q_3 , to signify odd states 1, 3, and 5 - all have an output of 1. You could then use another output state, say Q_2 , to code the family 4, 5, and 6 - all have an output of 0. These two lines then decode two of the base patterns without needing to use any logic gates.

	Q_1	Q_2	Q_3	Q'_1	Q'_2	Q'_3
6	0	0	0	1	1	1
4	1	0	0	0	1	1
2	1	1	0	0	0	1
1	1	1	1	0	0	0
3	0	1	1	1	0	0
5	0	0	1	1	1	0

Figure 1-8 Possible states for a modulo-6 switch tail ring counter

1-11 Given the truth table of states A, B, C, and D that you worked out in *question 1-1*, and the truth table above, how could you use a modulo-6 switch ring counter to define the 4 states of the die?

1.7 Exercise: Putting It All Together

Instructions:

- Build a 3-4 line encoder using logic gates.
 - Connect the appropriate inputs.
 - Connect the 7 probes in the "H" Pattern.
- **Start** the simulation.
- Verify that the simulation generates the dice patterns (1-6). If it does not, go back to the beginning and review:
 - Your understanding of the table outlining the possible states of the modulo-6 counter.
 - The 3 to 4 line encoder constructed with logic gates.
- **Stop** the simulation and observe what number is generated at the output.
- Try this again 5 more times to see which numbers are generated.

Record your observations below:

1-12 Is the number on the face of the die generated randomly? Explain.

1-13 How would you build this circuit if you wanted to simulate the rolling of two dice simultaneously to generate random numbers?

1.8 Exercise: Optional

As you have seen through some of the labs in this lab series, Multisim is an excellent tool for simulating circuits. You have also explored the ability to compile a circuit design onto the FPGA, and run a circuit with the peripherals on the board. Let's convert your circuit design to a real circuit.

- When running the circuit you'll need to use a real clock, such as the system clock, instead of a simulated clock.
- You could use real LEDs on the protoboard instead of the built-in LEDs, in order to make the "H" pattern of the dice.
- You also won't be able to just stop the simulation to signify when the random number is chosen, but you could use circuitry to write the current values of the number sequence to a flip-flop, which is triggered when you hit one of the buttons.
- Convert the rest of your design to work with these real components, compiled onto the FPGA.

Note: If you choose to complete this optional exercise, take a picture, draw a sketch, or take a screenshot of your final circuit and include it with your finished lab.

1.9 Conclusion

1-14 How many unique patterns exist on the face of 6-sided dice, from which all others can be formed?

- A. 2
- B. 4
- C. 5
- D. 6

1-15 In the modulo-6 counter, the inverted output of the last element is connected back to the input of the first element. This part of the circuit is called:

- A. An inverted loop
- B. Switch loop
- C. An inverted ring
- D. A switched tail ring

1-16 Why do we need to set the clock frequency of the dice several kilohertz?

- A. It increases the chances of the die stopping on a random number
- B. Less than the frequency would result in an incomplete spin
- C. At this frequency the eye cannot tell which numbers appear on the probes
- D. All of the above

1-17 When analyzing the table for the three to four line encoder, you see 3 Q' states. What does the ' symbol indicate?

- A. An odd state
- B. An even state
- C. An inverted state
- D. An identical output to the original

1-18 Implementing the dice on the Digital Electronics Board would require the following pieces of hardware:

- A. A real clock, LEDs, a flip-flop
- B. A real clock, resistors, LEDs
- C. Resistors, LEDs, a flip-flop
- D. None of the above

Lab Manual: Digital Electronics

Using the TI Digilent Digital Electronics Board for NI ELVIS III



Lab 16: Digital Clock – Application Lab #1

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 16: Digital Clock – Application Lab #1

In this lab, you will utilize a variety of concepts and skills developed in the digital electronics lab series to design and build a digital clock. The skills and concepts you will use include circuit design, counters, decoders, and seven-segment displays. The clock will consist of a series of counters driven by a pulse. There are a variety of extensions outlined in this lab that you may choose to include as part of your clock design.

Learning Objectives

In this lab, students will:

1. Design their own digital clock.
2. Explore how to connect counters in series to count seconds, minutes, and hours.
3. Learn how to display binary outputs on a seven-segment display.
4. Learn how to use a Function Generator to simulate clock pulse.
5. Have the opportunity to design and incorporate an alarm with a snooze button into their digital clock.
6. Have the opportunity to implement hardware on the Digital Electronics breadboard.

Required Tools and Technology

Platform: NI ELVIS III

Instruments used in this lab:

- Function Generator

Note: The NI ELVIS III Cables and Accessories Kit (purchased separately) is required for using the instruments

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M
- ✓ Install Soft Front Panel support:
<http://www.ni.com/documentation/en/ni-elvis-iii/latest/getting-started/installing-the-soft-front-panel/>

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>

Hardware: [Optional] Electronic Components

Components used in this lab:

- (1) BNC breakout plug

Optional Hardware:

12-hour clock:

- (1) green LED
- (1) 1000 Ω resistor
- Wire

LED Alarm:

- (1) red LED
- (1) 1000 Ω resistor
- Wire

Buzzer Alarm

- (1) 5 V buzzer
- (1) IN3064 Diode

- Wire

Seconds Display:

- (2) Seven-segment displays

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Software: NI LabVIEW FPGA Vivado 2014.4

- ✓ Install:
<http://www.ni.com/download/labview-fpga-module-2015-sp1/5920/en/>

Note: Digilent Driver (The installer above automatically downloads the installer below onto your computer)

- ✓ Navigate to:
C:\NIFPGA\programs\Vivado2014_4\data\xicom\cable_drivers\nt64\digilent
- ✓ Install: install_digilent.exe

Expected Deliverables

In this lab, you will collect the following deliverables:

- Definitions of key vocabulary terms
- Design decisions
- Roll-over table
- Video of functioning clock
- [Optional] Alarm block diagram sketch
- [Optional] Alarm sub-circuit screenshots
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

Digital Clock

Introduction Video



Figure 1-1 Video View the video here: <https://youtu.be/BNEM8n4qO5E>



Video Summary

- This video will explore how many of the concepts and skills you have learned in this course will help you design and build a digital clock
- Digital clocks are very commonly found in many everyday objects such as cell phones and microwaves
- This clock will have a series of counters that will be driven by a pulse

Clocks

Clocks, one of the oldest human inventions, play an important role in everyday life. All clocks, whether digital, analog or pendulum, need three basic capabilities to function. These are:

1. A power source to drive all the functions of the clock
2. A time base or pulse used to keep time
3. A way to display the time

The two most common types of clocks are digital and analog. The difference between these two clocks is how they display the time. An *analog clock*, Figure 1-2, indicates the time using a set of rotating hands on a labeled surface whereas a *digital clock*, Figure 1-3 displays a series of LED digits that correspond to the current time. Digital clocks are common because they are inexpensive, compact and easy to read, which allows for them to be used in many different applications such as cars, televisions, and microwaves.

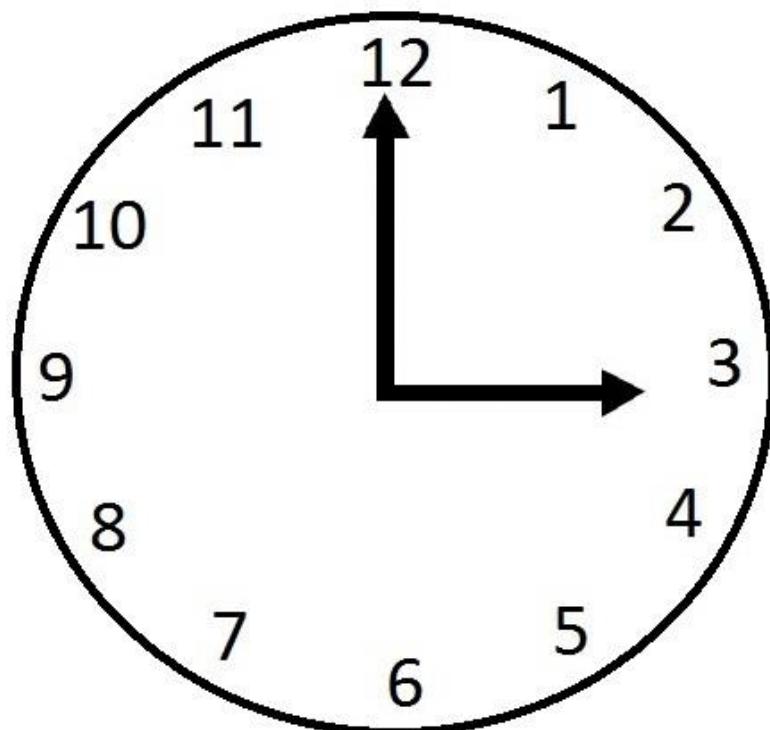


Figure 1-2 Analog clock



Figure 1-3 Digital clock

Counters

In this lab, you will build a digital clock. The clock will consist of a series of counters driven by a pulse. The pulse will drive the first counter, which will drive the second counter, which will then drive the third, and so on.

A Multisim seven-bit counter chip is shown below:

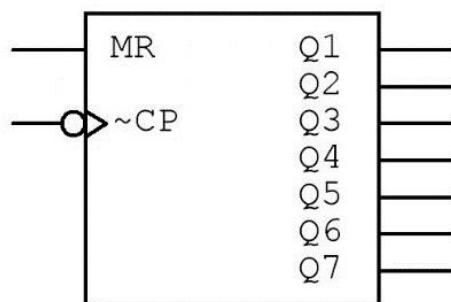


Figure 1-4 Seven-bit counter chip

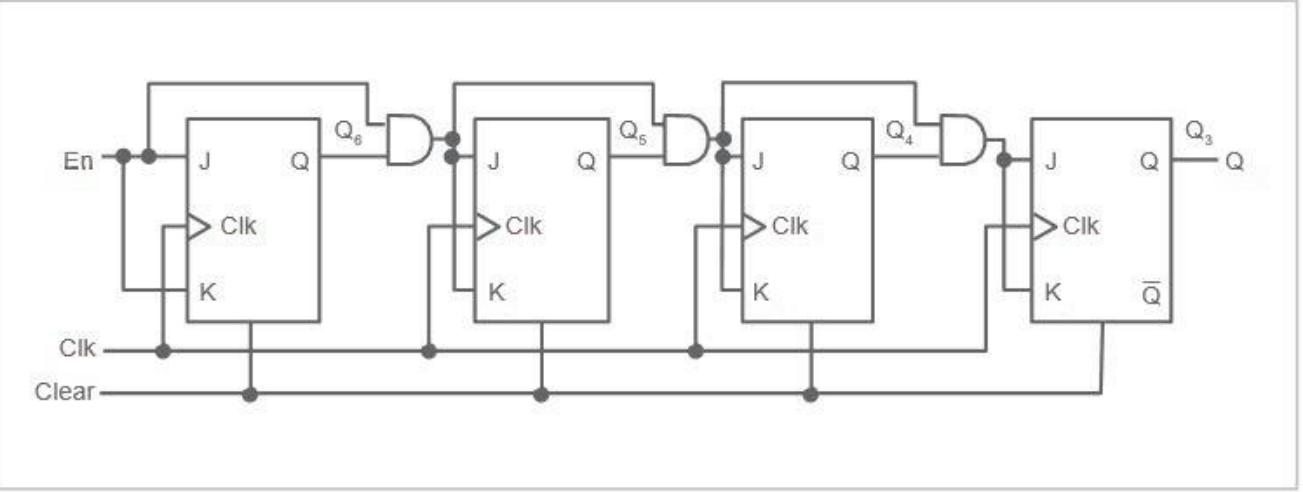


Figure 1-5 Series of four-bit counters

Seven Segment Display on the Digital Electronics Board

Seven-segment displays are electronic displays, commonly used by digital clocks that display decimal numerals. Each display contains seven individual LED units which create the seven different segments. These segments are named "a" through "g" following a clockwise pattern, as shown in the figure below, and together can form the numbers zero through nine.

The outputs of the counters in your clock circuit will be in binary, so to communicate information between the counters and the displays a *decoder* needs to be added between them. This decoder will take the binary values from the counter and convert them into the corresponding combination of segments on the display. For example, the decoder would take a binary input of 0010 (decimal 2) and output 1s to segments a, b, g, e, and d to form the number 2 in the display.

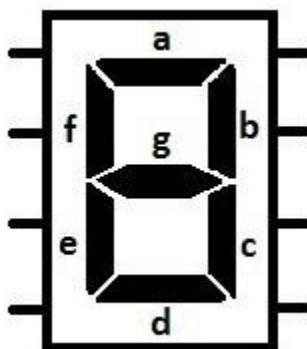


Figure 1-6 Seven segment display

Using the Digital Electronics Board SSD

Since FPGA chips are reconfigurable, they do not have 28 different output pins for the 4 SSD digits, in view of efficiency. Instead, they only have one set of 7 pins to control each segment of the SSD and another 4 pins to select which digits are turned on.

To simultaneously display a different number on each SSD digit, it is common practice to sequentially alternate between displaying each digit and change the data provided to the 7 pins accordingly as each next digit is turned on. If the alternation is carried out quickly enough, each digit will appear to be turned on continuously.

In this lab, you will use the technique described above. You will alternate between each digit at a frequency of 1024 Hz to ensure the digits appear to be turned on continuously.

Buffers

You will be using *tristate buffers* in this lab. Tristate buffers have one data input, one enable input and an output.

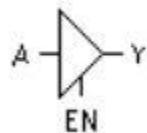


Figure 1-7 Tristate buffer

The truth table of the tristate buffer is provided below:

Input	Enable Input	Output
1	1	1
0	1	0
X	0	Z

When the enable input is zero, the buffer enters a *high-impedance (Z) state* regardless of the input. In this state, the device disconnects the rest of the circuit from the output pin. Therefore, the tristate buffer is commonly used for circuits with nodes that receive outputs from different digital devices.

In addition, you will be using the *NI ELVIS III Function Generator* in this lab. The system clock of the Digital Electronics Board is set at a frequency of 128 MHz, but this

frequency is fixed. The function generator allows you to freely control the input frequency, hence changing how fast the clock runs.

Before starting this application lab, it is suggested that you review the following labs:

- Lab 6: Encoders and Decoders
- Lab 10: Flip-Flops (for 12-hour clock only)
- Lab 11: Counters
- Lab 13: Shift Registers

Research and Planning

- Research and summarize the following keywords to help you gain a better understanding of what you will need in this lab.
- You will be able to come back and add more information at any point in the lab.

1-1 Digital clock definition:

1-2 12-hour clock definition:

1-3 24-hour clock definition:

1-4 Counter definition (binary or decimal output, clock and clear functions, etc.) :

1-5 Seven-segment display definition (pinout, wiring instructions, etc.) :

1-6 BCD to seven-segment display decoder definition (pinout, function, etc.) :

1-7 Function generator definition:

- Answer the following questions to help guide you through your brainstorming and design process. These questions will also ensure your research covered the necessary information before you start building your clock.

1-8 Decide if you want to build a 12-hour or 24-hour clock. Explain why you have made this decision and describe the unique challenges that you will face.

1-9 Make a checklist of the different things your circuit should be able to do.

1-10 All the counters you will use for this lab are binary output counters. For this project, they will need to count up to a maximum of nine and then reset to zero. How many input and output pins will the counters need? How many bits will be needed to count that amount?

1-11 In this lab, you use the Function Generator to control the frequency input of your clock. How will the Function Generator be beneficial during the testing phase of your clock, compare to just using the system clock?

1-12 How many seven-segment displays are you going to use?

1-12a What do the lab instructions require?

1-12b How many SSDs are available to you on the Digital Electronics Board?

1-12c What does a typical clock show?

1-13 How high does each counter need to count before it rolls over?

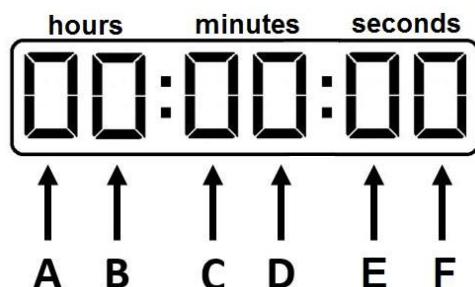


Figure 1-8 Counter

Digit	Number Before Roll-over
A	
B	
C	
D	
E	
F	

1.2 Exercise: Building the Digital Clock

Follow the guidelines provided in this step to build your digital clock circuit. If you need additional help troubleshooting problems, see the *Testing and Troubleshooting* Section.

Use the following block diagram to help you build your clock design. The addition for the optional 12-hour clock is shown with dashed lines.

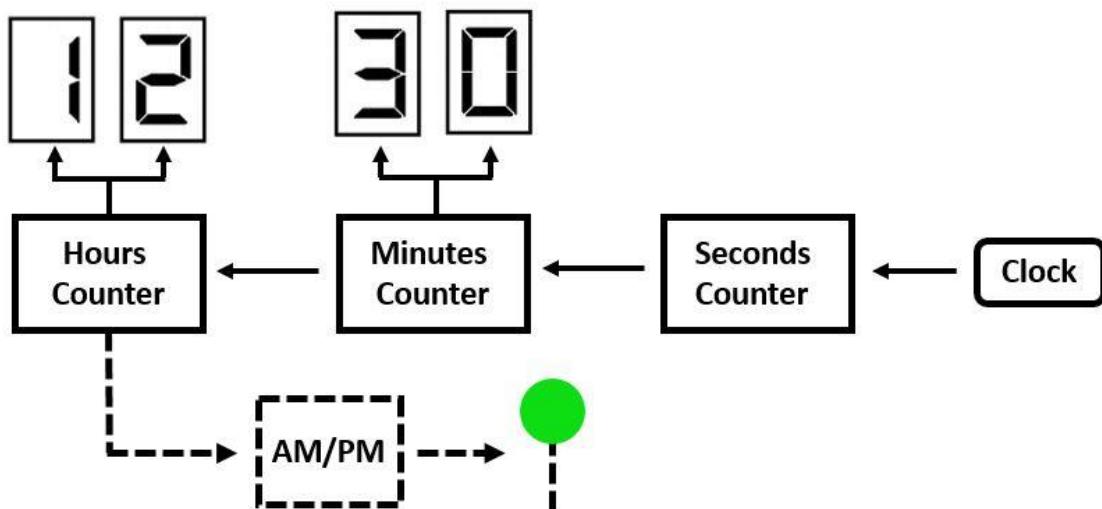


Figure 1-9 Clock design block diagram

Tip: Try adding both FPGA controls and interactive digital constants, as well as probes, so that you can test the different components of your circuit virtually without having to export to the Digital Electronics Board.

1. Open a **NEW** PLD design.
2. Create three different subscripts named **Seconds**, **Minutes** and **Hours**.
3. Place the PLD input connector **BB_S_DIO4** in the main PLD design.
4. Place the seven PLD output connectors for the seven-segment displays **SSEG_CA** to **SSEG_CG** in the main PLD design.

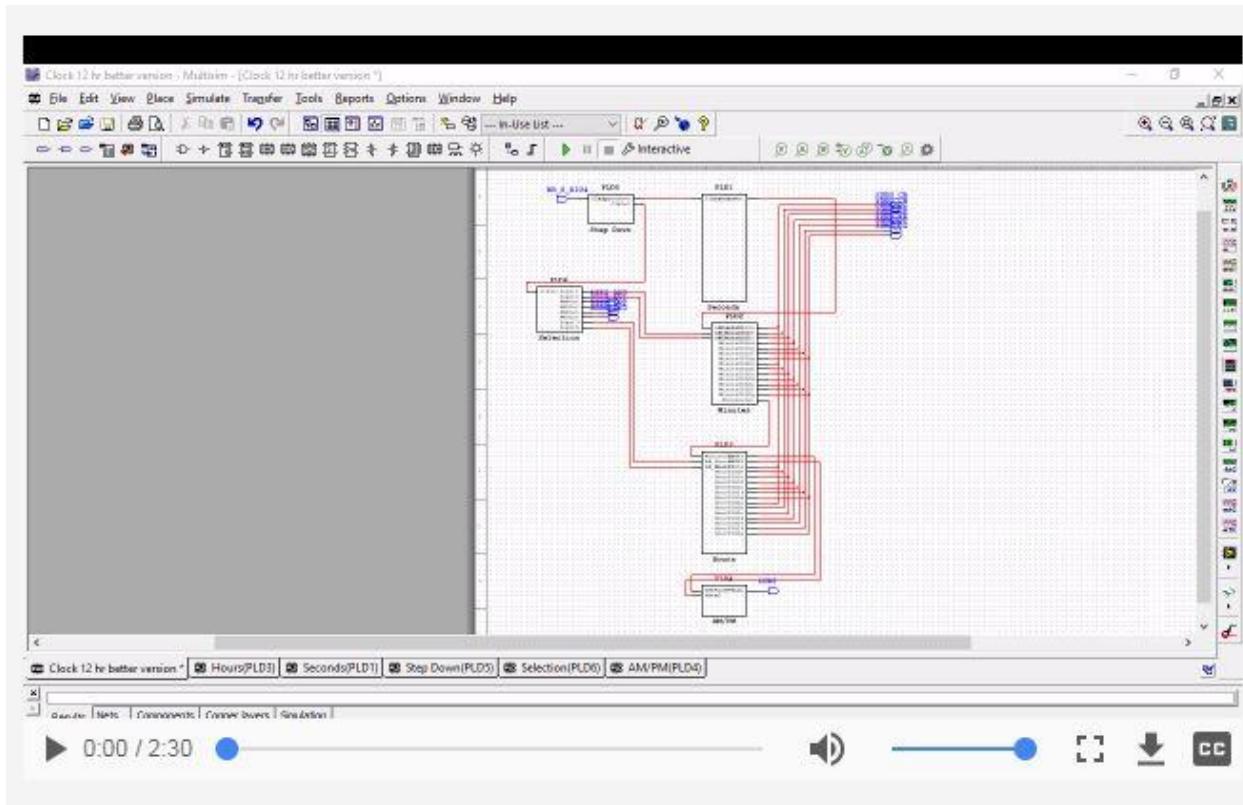


Figure 1-10 Circuit overview video View the video here: https://cf-ts.mythinkscape.com/video/Clock - General_no_cursor_2.mp4

Components Overview

- Seconds Sub-circuit:
<https://mythinkscape.com/labs/v2/23200/steps/21139#Seconds%20SubC>
- Minutes Sub-circuit:
<https://mythinkscape.com/labs/v2/23200/steps/21139#Minutes%20SubC>
- Hours Sub-circuit:
<https://mythinkscape.com/labs/v2/23200/steps/21139#Hours%20SubC>
- Step-Down Sub-circuit:
<https://mythinkscape.com/labs/v2/23200/steps/21139#Step-Down%20SubC>
- Selection Sub-circuit:
<https://mythinkscape.com/labs/v2/23200/steps/21139#Selection%20SubC>

Seconds Sub-circuit

1. In the **Seconds** sub-circuit, place the following components:
 - (2) 7-bit binary counters (CNTR_7BIN)

Note: These counters will allow you to count up to the desired number then reset at 0.

- (2) AND4 gates and (4) inverters

Note: They will ensure the counters roll over at the desired time.

2. Add (1) PLD input connectors and name it **ClockPulse**.
 3. Add (1) PLD output connectors and name it **SecondsOut**.
 4. Once all the components have been placed in the workspace, wire the Seconds sub-circuit following the diagram below.

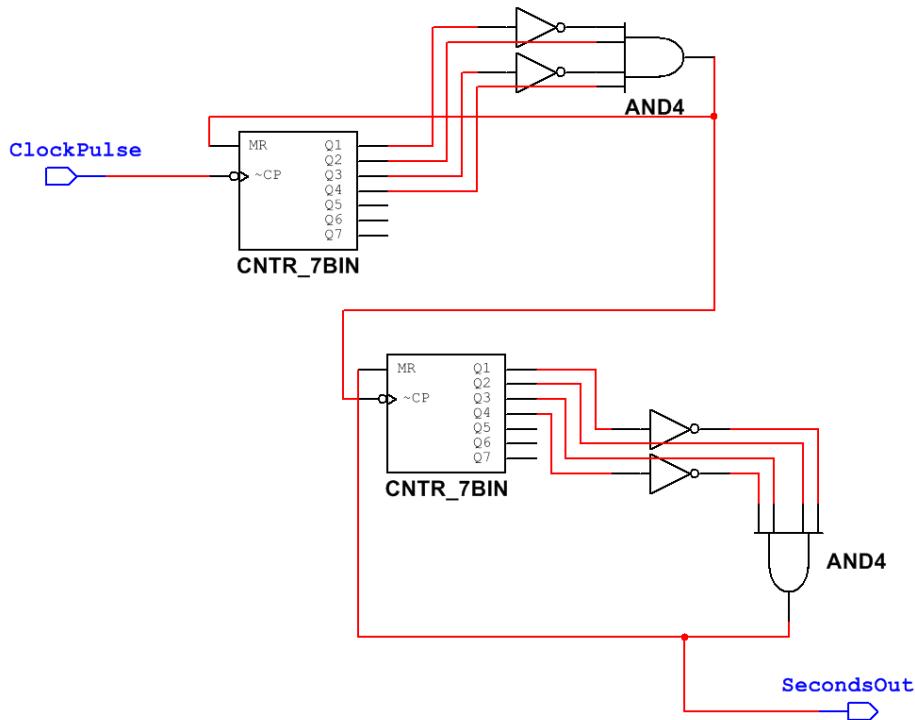


Figure 1-11 Seconds sub-circuit

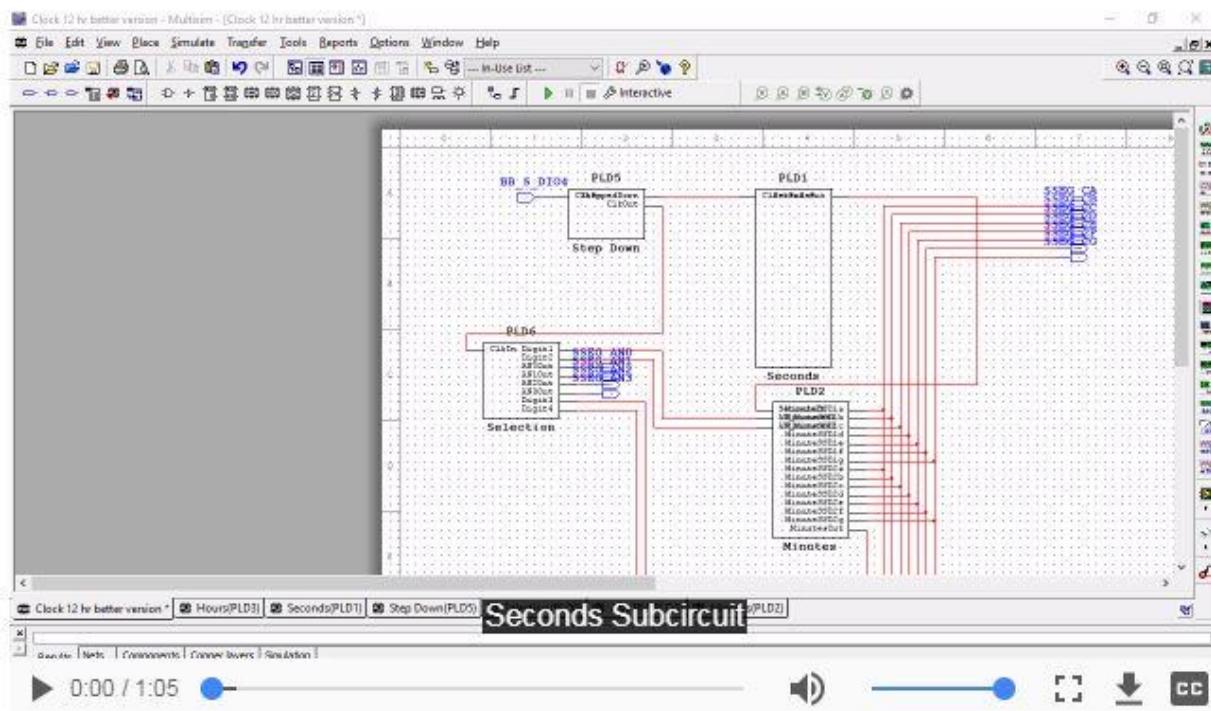


Figure 1-12 Seconds sub-circuit video View the video here: https://cf-ts.mythinkscape.com/video/Seconds_Subcircuit_no_cursor.mp4

Minutes Sub-circuit

1. In the **Minutes** sub-circuit, place the following components:
 - (2) 7-bit binary counters (CNTR_7BIN)

Note: They will allow you to count up to the desired number then reset at 0.

- (2) BCD to SSD decoders (DEC_BCD_7)

Note: They convert the binary counter output into data that the seven-segment display can read and display.

- (2) digital high

Note: They are needed for the decoders to run.

- (2) AND4 gates and (4) inverters

Note: They will ensure the counters roll over at the desired time.

- (14) tristate buffers (BUF_3S_AHOE)

Note: They send the correct information to the SSD output pins when the corresponding digit is being updated.

2. (3) PLD input connectors and name them:

- **SecondsIn**
- **AN_Minutes1** and **AN_Minutes2**

Note: They enable the correct digit on the board (the two digits on the right) to be updated. See the Selection sub-circuit for more information.

3. Add (15) PLD output connectors and name them:

- **MinuteSSD1a** to **MinuteSSD1g**
- **MinuteSSD2a** to **MinuteSSD2g**
- **MinutesOut**

4. Once all the components have been placed in the workspace, wire the Minutes sub-circuit as shown in the diagram below.

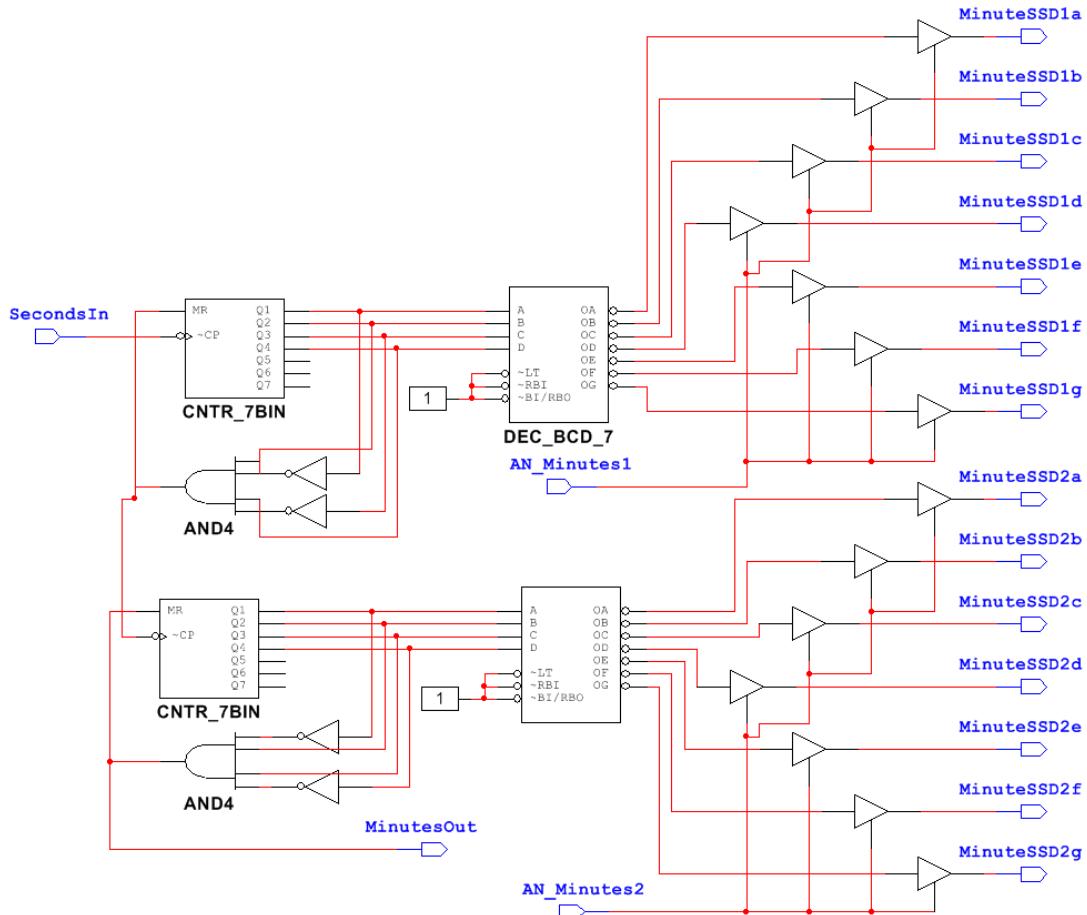


Figure 1-13 Minutes sub-circuit

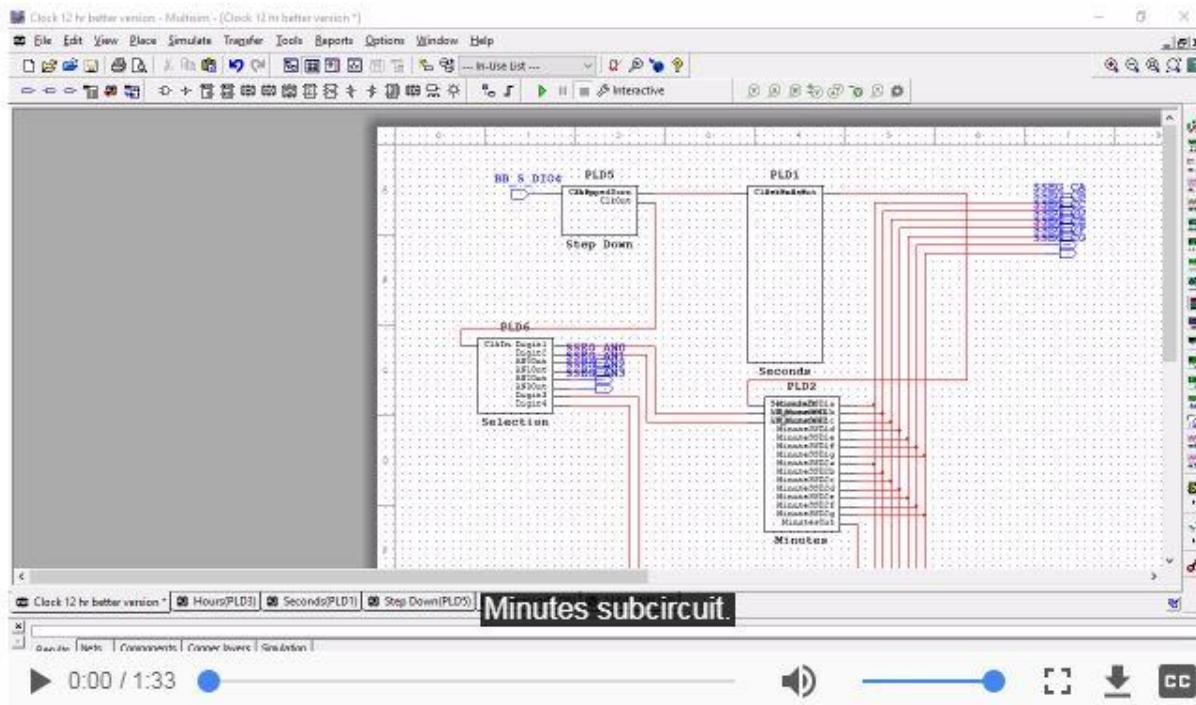


Figure 1-14 Minutes sub-circuit video View the video here: https://cf-ts.mythinkscape.com/video/Minutes_Subcircuit_no_cursor.mp4

Hours Sub-circuit

You can choose to build a 24-hour or 12-hour clock.

- Click: <https://mythinkscape.com/labs/v2/23200/steps/21139#24-Hour%20SubC> for a 24-hour clock
- Click: <https://mythinkscape.com/labs/v2/23200/steps/21139#12-Hour%20SubC> for a 12-hour clock and <https://mythinkscape.com/labs/v2/23200/steps/21139#Am/PM%20Display> for the AM/PM display

24-Hour Clock Sub-circuit

1. In the **Hours** sub-circuit place the following components:
 - (2) 7-bit binary counters (CNTR_7BIN)

Note: They will allow you to count up to the desired number then reset at 0.

- (2) BCD to SSD decoders (DEC_BCD_7)

Note: They convert the binary counter output into data that the seven-segment display can read and display.

- (2) digital high

Note: They are needed for the decoders to run

- (1) AND4 gate, (2) inverters, (1) AND gate and (1) OR gate

Note: They will ensure the counters roll over at the desired time.

- (14) tristate buffers (BUF_3S_AHOE)

Note: They send the correct information to the SSD output pins when the corresponding digit is being updated.

2. (3) PLD input connectors and name them:

- MinutesIn
- AN_Hours1 and AN_Hours2

Note: They enable the correct digit on the board (the two digits on the left) to be changed. See *the Selection sub-circuit* for more information.

3. Add (14) PLD output connectors and name them:

- HourSSD1a to HourSSD1g
- HourSSD2a to HourSSD2g

4. Once all the components have been placed in the workspace, wire the Hours sub-circuit as shown in the diagram below.

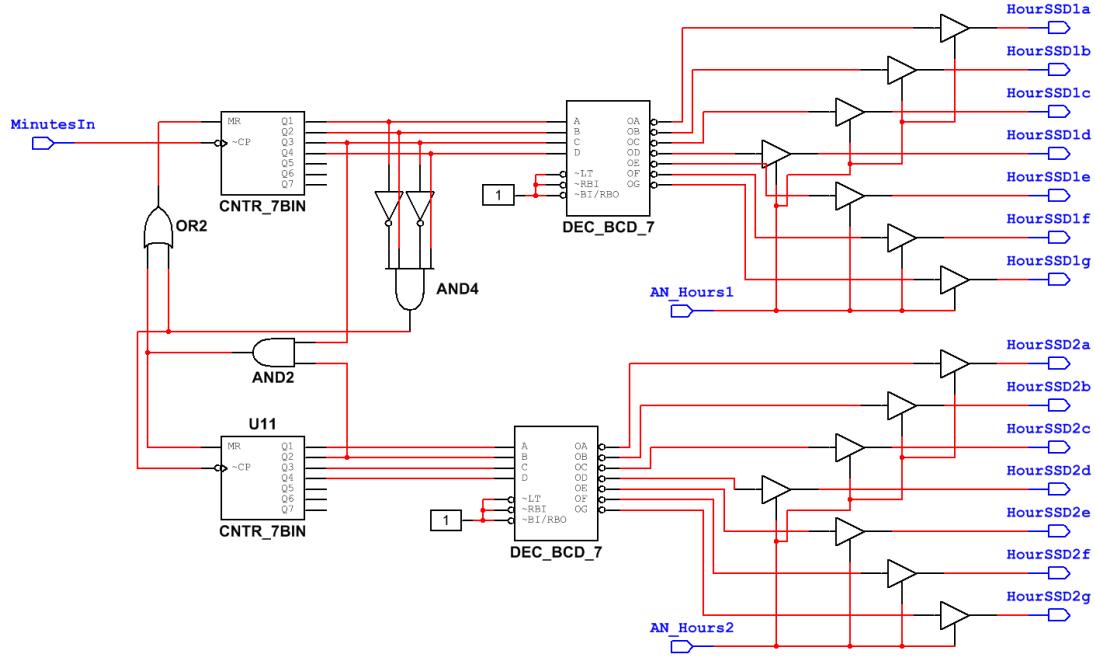


Figure 1-15 24-hour sub-circuit

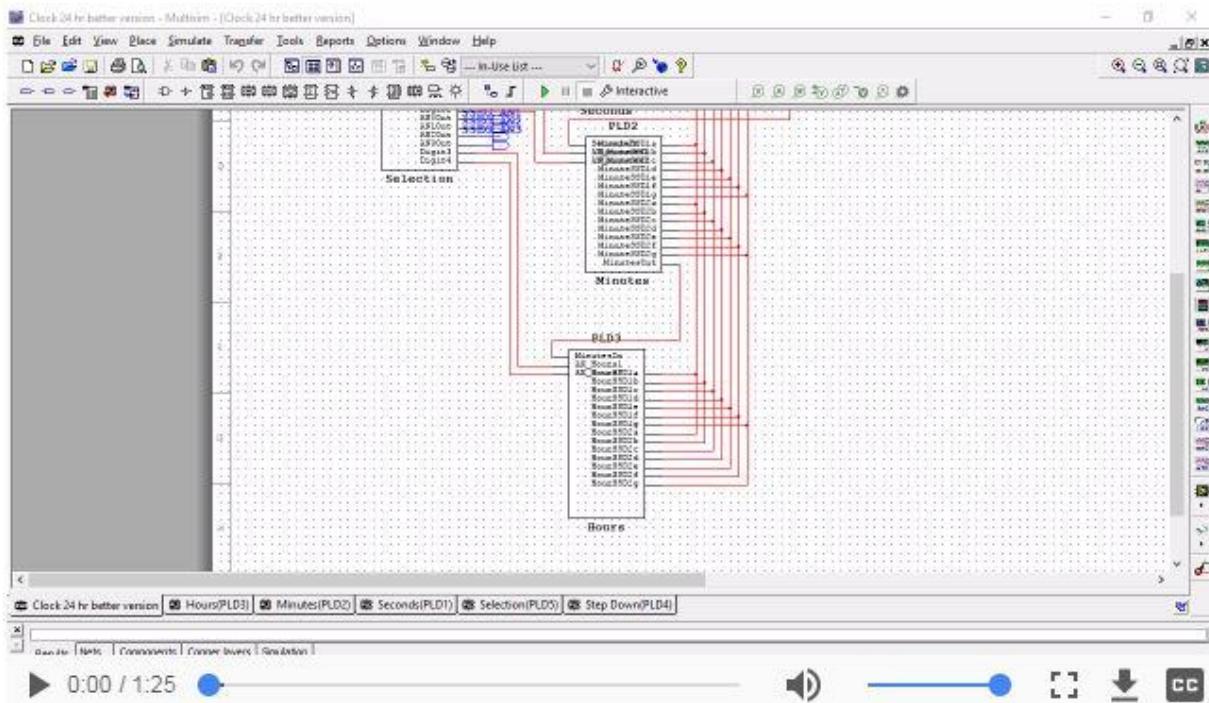


Figure 1-16 24-hour sub-circuit video View the video here: https://cf-ts.mythinkscape.com/video/Hours_Subcircuit_24H_no_cursor.mp4

When you are done, follow the link here:

<https://mythinkscape.com/labs/v2/23200/steps/21139#Step-Down%20SubC>.

12-Hour Clock Sub-circuit and AM/PM Display

1. In the **Hours** sub-circuit place the following components:
 - (2) 7-bit binary counters (CNTR_7BIN)

Note: These counters will allow you to count up to the desired number then reset at 0.

- (2) BCD to SSD decoders (DEC_BCD_7)

Note: These decoders convert the binary counter output into data that the seven-segment display can read and display.

- (2) digital high

Note: This is needed for the decoders to run.

- (1) AND2 gate, one AND4 gate, (2) inverters and (2) OR2 gates

Note: These gates ensure the counters roll over at the desired time.

2. (3) PLD input connector and name them:
 - **MinutesIn**
 - **AN_Hours1** and **AN_Hours2**
3. ADD (16) PLD output connectors and name them:
 - **HourSSD1a** to **HourSSD1g**
 - **HourSSD2a** to **HourSSD2g**
 - **AMPM1** to **AMPM2**
4. Once all the components have been placed in the workspace, wire the Hours sub-circuit as shown in the diagram below.

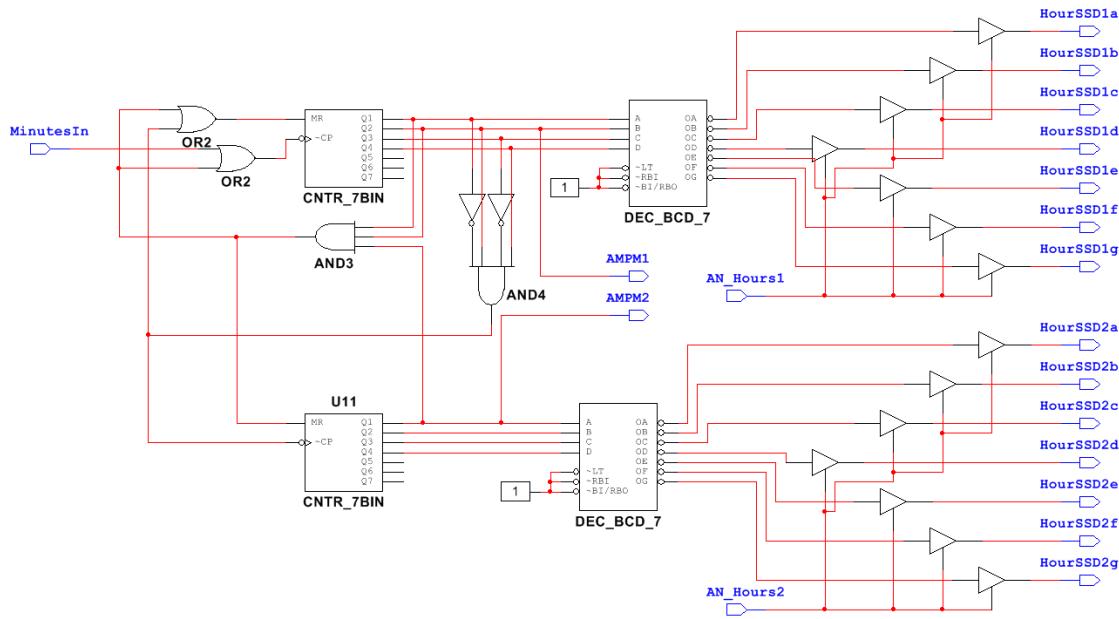


Figure 1-17 12-hour sub-circuit

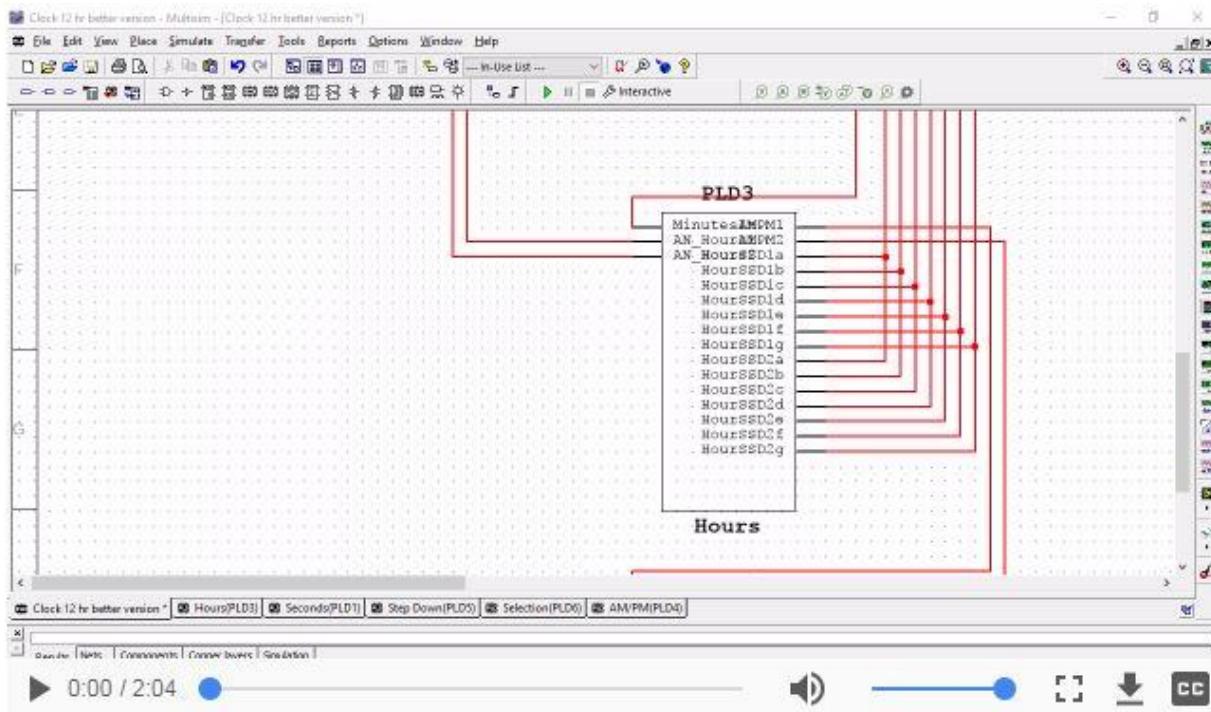


Figure 1-18 12-hour sub-circuit video View the video here: https://cf-ts.mythinkscape.com/video/Hours_Subcircuit_12H_no_cursor.mp4

The AM/PM Display

1. Create a fourth sub-circuit in the main PLD design and name it **AMPM**.
 2. In this sub-circuit, place the following components:
 - o (1) D flip-flop

Note: This flip-flop holds a value so that the LED remains on for the complete cycle

- o (1) inverter

Note: This gate resets the flip-flop when it changes from AM to PM

- (1) AND gate

Note: This gate ensures the LED changes between 11 and 12.

3. Add (2) PLD input connectors and name them: **AMPM1** and **AMPM2**.
 4. Add (1) PLD input connector and name it **AMPMLED**.
 5. Once all the components have been placed in the workspace, wire the AM/PM Display circuit as shown in the diagram below.

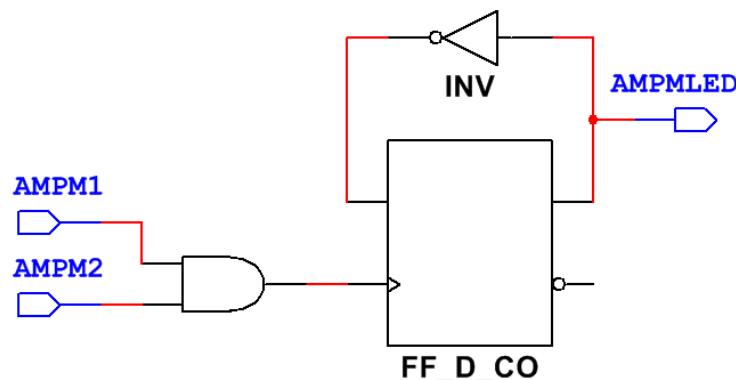


Figure 1-19 AM/PM Sub-circuit

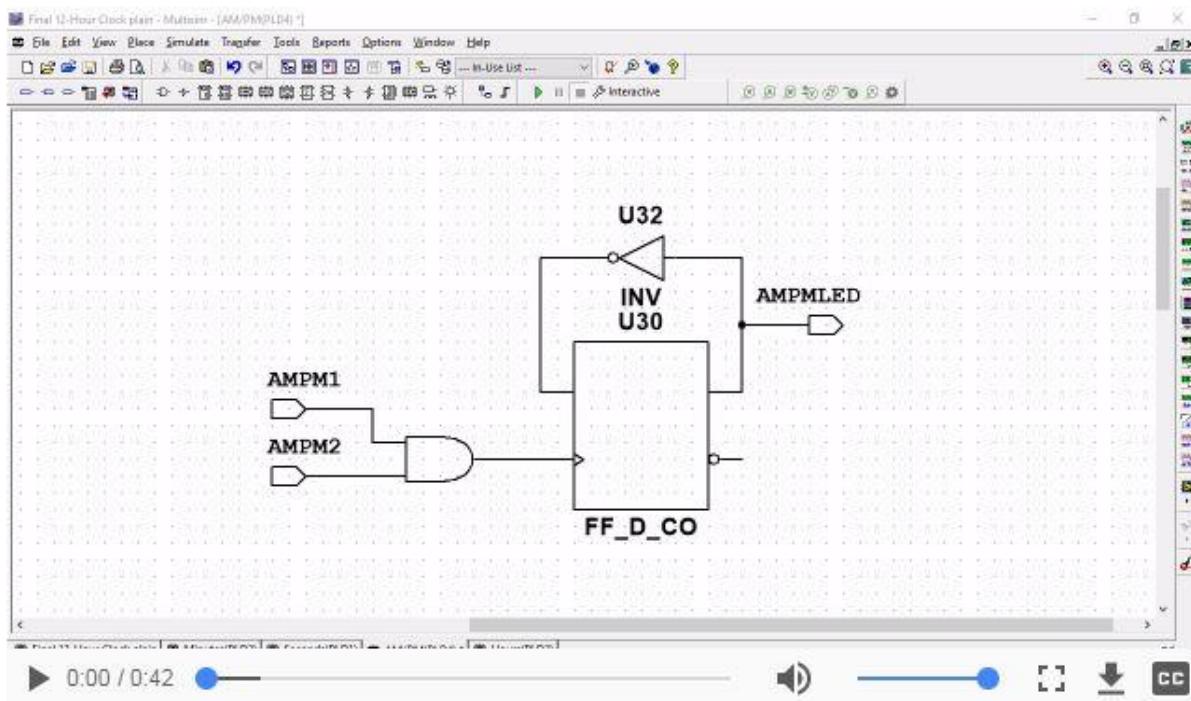


Figure 1-20 AM/PM display Sub-circuit video View the video here: https://cf-ts.mythinkscape.com/video/AMPM_no_cursor.mp4

Step-Down Sub-circuit

Recall from the lab introduction that you will use a 1024 Hz frequency to update each SSD digit's value to make them appear continuously turned on. However, you must step down this frequency to 1Hz before it can act as a clock pulse to the Seconds sub-circuit. This is the function of the Step-Down sub-circuit.

1. Create a sub-circuit in the main PLD design and name it **Step-Down**.
2. Place the following components in this sub-circuit:
 - (3) 4-Bit Asynchronous Binary Counters (CNTR_4BIN_AS)

Note: They are used to output a high signal once every fixed number of input clock pulses, hence decreasing the clock frequency.

- (3) digital low

Note: They prevent the counters from resetting

3. Add (1) PLD input connector and name it **ClkIn**
4. Add (2) PLD output connectors and name them:

- **ClkOut**
 - **SteppedDown**
5. Once all the components have been placed, wire the Step-down sub-circuit as shown in the diagram below.

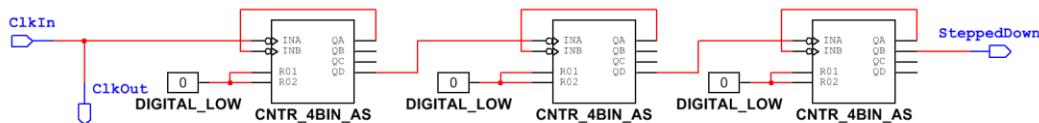


Figure 1-21 Step-down sub-circuit

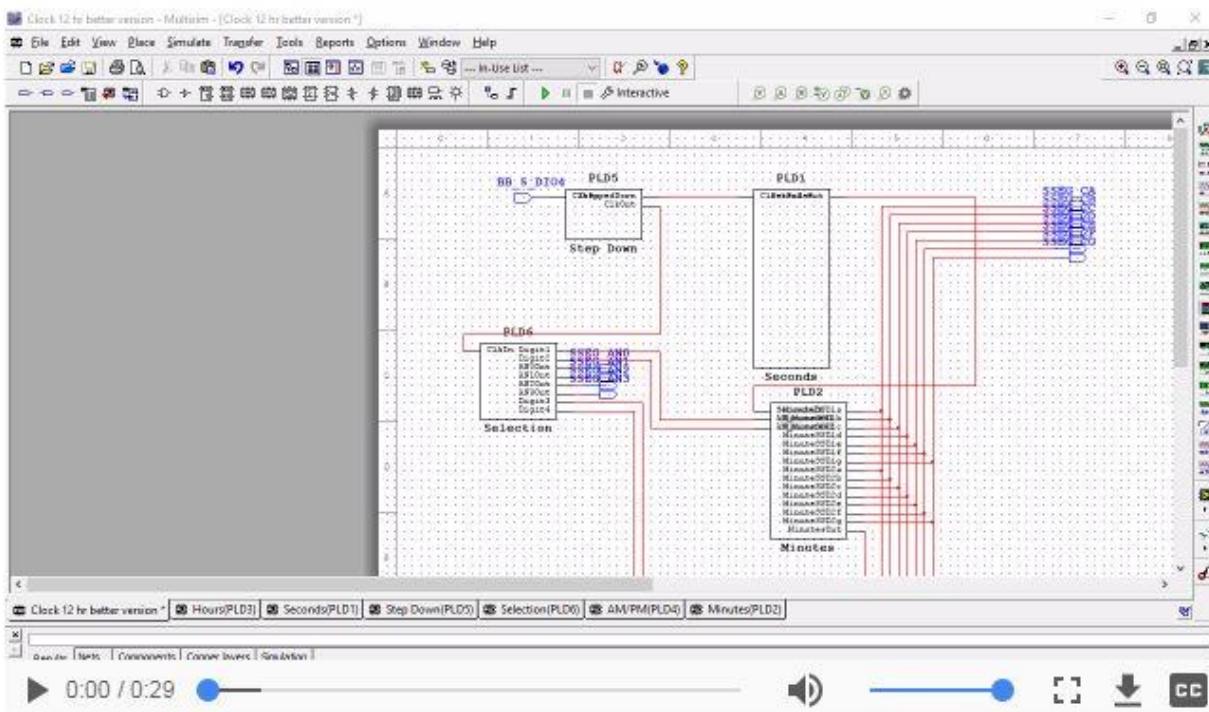


Figure 1-22 Step-down sub-circuit video View the video here: https://cf-ts.mythinkscape.com/video/Stepdown_Subcircuit_no_cursor.mp4

Selection Sub-circuit

The function of the selection sub-circuit is to sequentially update the value of each SSD digit, as described in the *Step-Down Sub-circuit*.

1. Create a sub-circuit in the main PLD design and name it **Selection**.
2. Place the following components in this sub-circuit:
 - o (1) 4-bit Asynchronous Binary Counter (CNTR_4BIN_AS)

- It provides a signal to the data input of the shift register every 5 clock pulses.
 - (1) 4-Bit SIPO Shift Register (SR_4S_P)
 - It sends a high signal sequentially to each of the four digits, hence allowing the digits to update at regular, high-frequency intervals.
 - (1) digital low
 - (1) AND gate and four inverters
3. Place (1) PLD input connector and name it **ClkIn**.
4. Place (8) PLD output connectors and name them:
- **Digit1** to **Digit4**
 - **AN0Out** to **AN3Out**
5. Once all the components have been placed, wire the Step-down sub-circuit as shown in the diagram below.

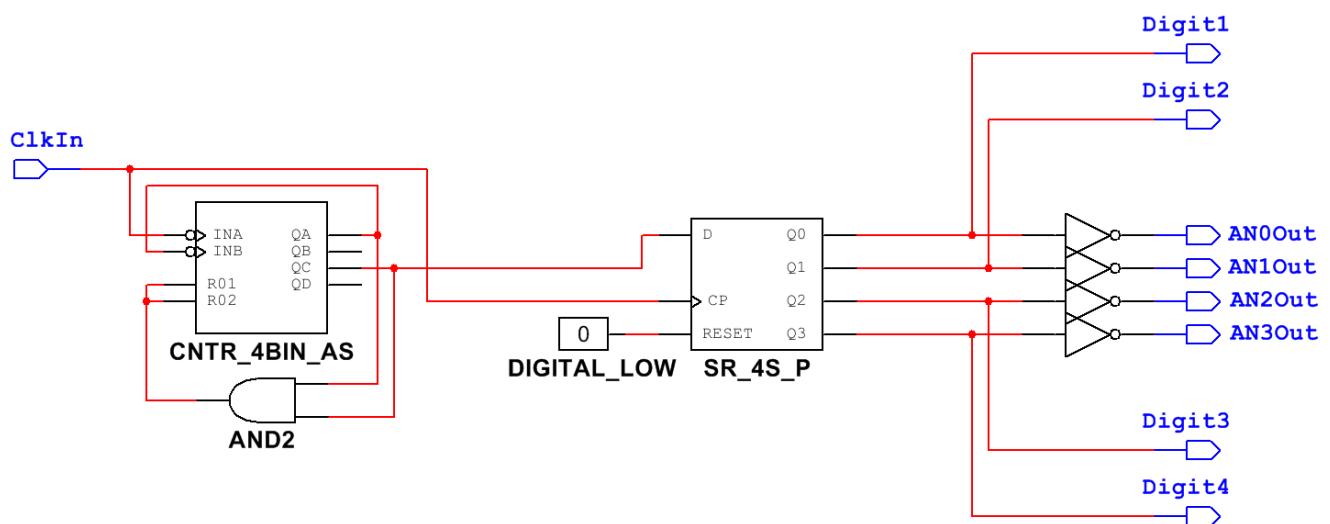


Figure 1-23 Selection sub-circuit

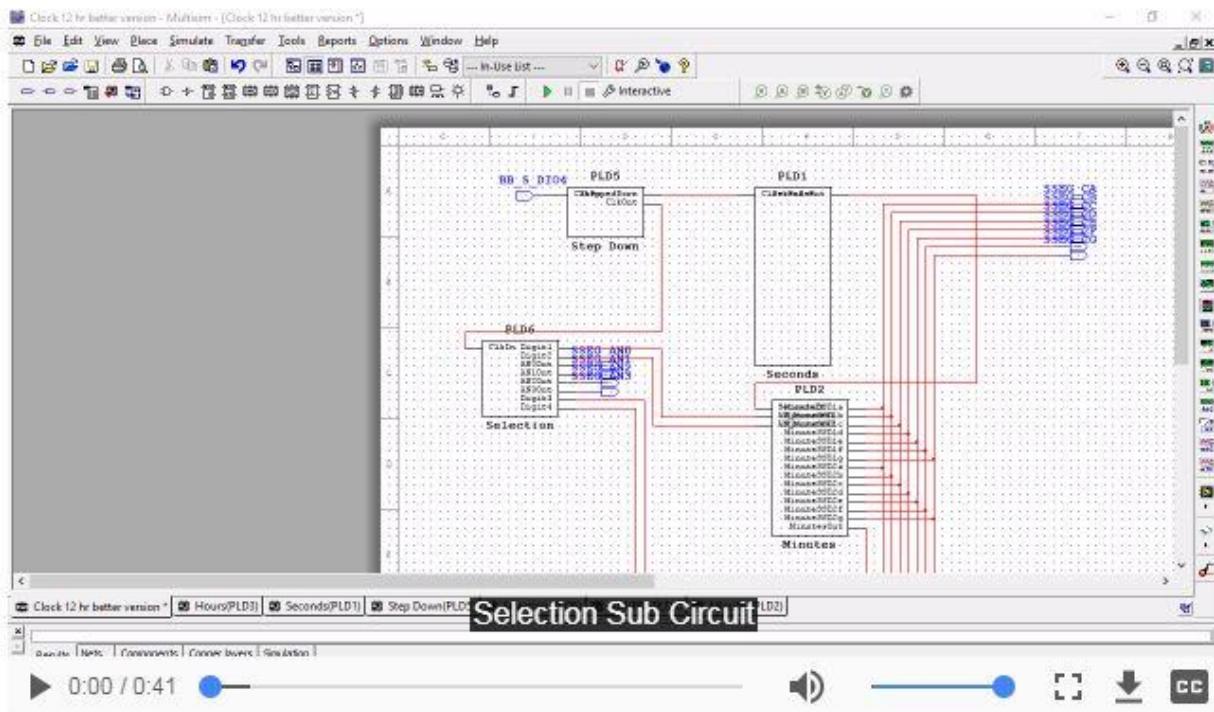


Figure 1-24 Selection sub-circuit video View the video here: https://cf-ts.mythinkscape.com/video/Selection_no_cursor.mp4

1.3 Exercise: Setting Up the Function Generator

Instructions:

1. Connect the BNC plug to the FGEN port of the ELVIS platform. Connect the red clip to a wire and plug the wire into the **BB_S_DI04**. Similarly, connect the black clip to a wire and plug it into the **GND**.
2. Set the following values to the Function Generator:
 - o Waveform: **Square**
 - o Frequency: **1,024 Hz**
 - o Amplitude: **3.00 Vpp**
3. Make sure that the Device (**NI Elvis III**) and Signal Route (**FGEN BNC**) are successfully recognized by the Function Generator. In order to be recognized, the Elvis Board must be turned on.

Use the following image as a reference for the settings:

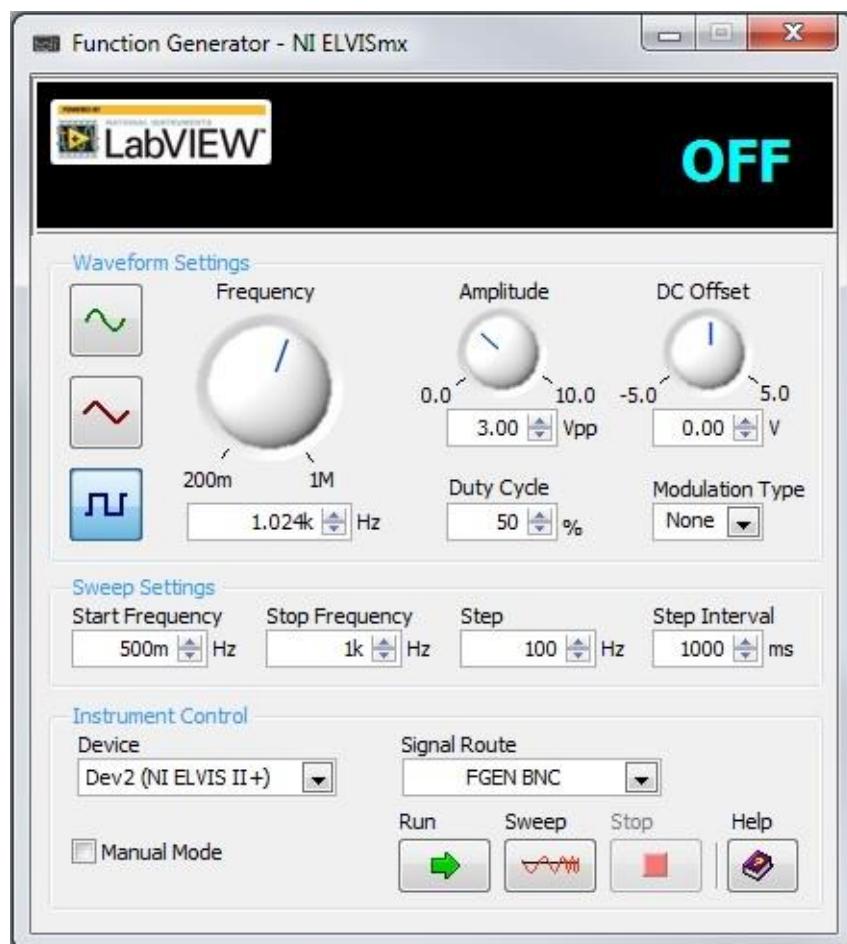


Figure 1-25 Settings reference

1.4 Implement: Putting It All Together

1. Place (4) PLD output connectors **SSEG_AN0** to **SSEG_AN3** in the main PLD design.
2. Connect all of the sub-circuits as shown in the figure below to complete your digital clock.
 - o For 24-hour clock:

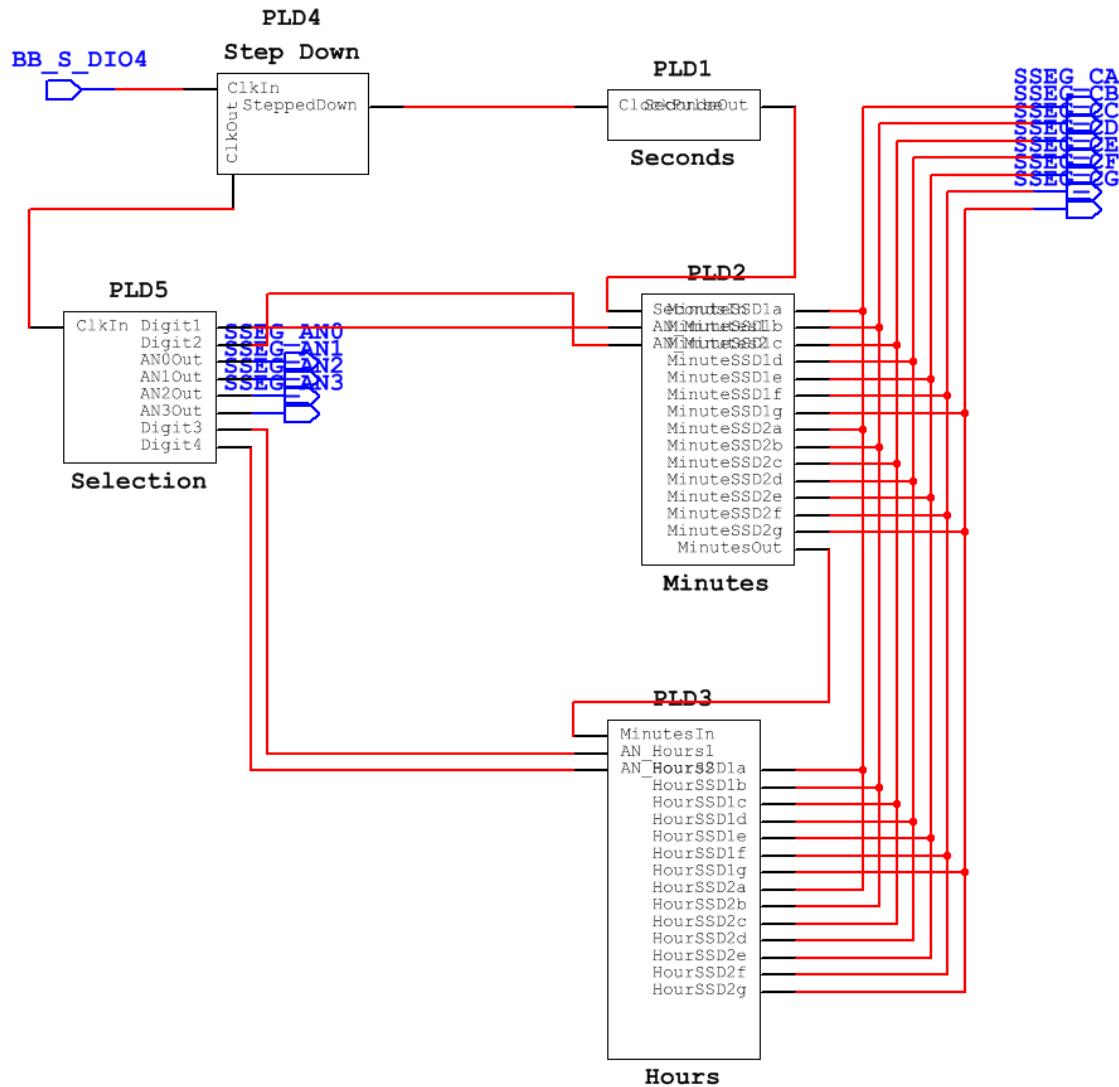


Figure 1-26 Connected sub-circuits for 24-hour clock

- For 12-hour clock:
 - Place the PLD output connector **LED0** in the main PLD design and connect your circuit.

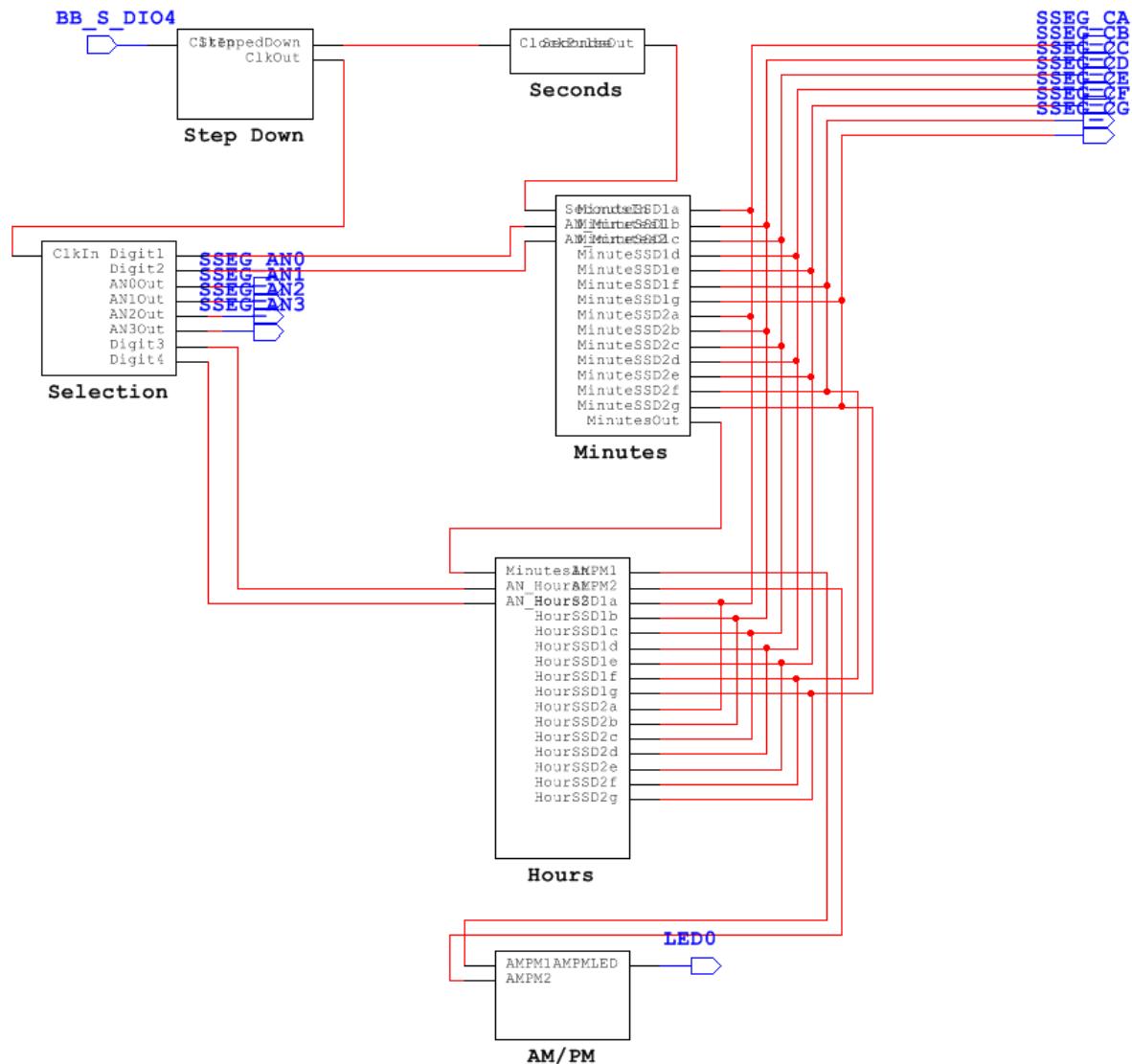


Figure 1-27 Connected sub-circuits for 12-hour clock

3. Deploy the circuit to the Digital Electronics Board.

Note: Take a screenshot, draw a sketch, or take a picture of your finished circuit in PLD as well as the individual subscripts and include them with your finished lab.

1.5 Testing and Troubleshooting

Go through the following steps to test that your clock is working properly. If you come across any issues, use the questions later in this section to help you troubleshoot and fix mistakes.

Tip: Increase the frequency of your source to speed up the testing process.

- What frequency have you chosen for testing?
- What frequency should the source be if you want your clock to work normally?

Instructions:

1. Export the circuit to the Digital Electronics Board.
2. Let your clock run for 2 minutes. Make sure that it is not running too fast or too slow.
3. Let your clock run through a full accelerated 24-hour cycle. Make sure the minutes and hours are changing properly.
4. **12-hour clock:** make sure your AM/PM display is working correctly. It should transition from AM to PM between the 11th and 12th hours.

Use the following questions to help you troubleshoot and fix any issues with your circuit.

Things to always check:

- Did you turn your Digital Electronics Board off and on before deploying it?
 - The board needs to be turned off and on to reset

The PLD circuit won't deploy to the Digital Electronics Board / hasn't deployed properly.

- Ensure you have turned on your NI ELVIS III and your Digital Electronics Board.
- Be sure to select the correct board when prompted.
- Check that the correct connectors have been chosen. They must match up with those on the Digital Electronics Board.
- Double check you have correctly wired the circuit on the board.

The minute display isn't increasing for 0 to 59 properly.

- Check that the seconds counters are increasing from 0 to 59 properly.
- Check the connection between the seconds counters and the minute counters.
The minute counter should increase by 1 when the seconds counter displays 59.
- Check the connections between the two minute counters are correct. They should be wired as shown in Figure 1-13.

- Check that the binary output of the counter is being properly converted so that it can be correctly displayed on the seven-segment display.

The minute display isn't returning to 0 after 59.

- Check that the right counter values are resetting your minute counters. The values output by the counter are in binary. Use the following table to ensure that you have selected the proper values. This part of the circuit should resemble the figure in the Building section.

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Figure 1-28 Decimal to binary conversion table

The hour display isn't increasing from 0-24 or (1-12) properly.

- Check the connection between the two, hour counters. This circuit should resemble the figure in the *Building* section.
- Check that the decoder is properly connected and is converting from binary to SSD correctly.

The hour display isn't returning to 0 after 23:59.

- Check that the right counter values are resetting your hour counters. Use *Figure 1-28 Decimal - Binary Conversion Table* to ensure your counter is outputting the correct numbers.

12-Hour Clock:

The AM/PM function is changing at the wrong time.

- Make sure that the flip-flop is not changing or clearing when it shouldn't.
- Make sure that you are using the correct output from the hour counters.

The AM/PM function isn't changing at all, or it's only changing once.

- Make sure that your counter and or flip-flop are clearing properly.
- Make sure that your counter and or flip-flop are connected to the clock properly.

1.6 Extensions

This lab contains four extension activities. Choose *ONE or more* of these extensions to add to your clock:

- Seconds display:
<https://mythinkscape.com/labs/v2/23200/steps/21146#Seconds%20Display>
- Setting display time:
<https://mythinkscape.com/labs/v2/23200/steps/21146#Setting%20Display%20Time>
- Alarm: <https://mythinkscape.com/labs/v2/23200/steps/21146#Alarm>
- Alarm with a buzzer or an LED:
<https://mythinkscape.com/labs/v2/23200/steps/21146#Alarm%20with%20Buzzer%20or%20LED>

Seconds Display

The following section discusses an addition to the functionality of the clock to display the seconds.

It's not uncommon for a clock to display more detail than just the hours and minutes. In order to do this, you will need to convert values 0-59 seconds into inputs for two new seven-segment displays.

Answer the following questions to help guide you in your brainstorming and design process for the setting display time option:

1-14 The Digital Electronics Board only has 4 on board seven-segment displays. How would you connect additional displays?

1-15 How would you output the signals from the FPGA chip to control the displays?

Use the following guidelines to add this extension to the digital clock you built in the previous section, following the plans you just made.

1. Modify the **Seconds** sub-circuit in order to display 2 digits to seven-segment displays.

Testing and Troubleshooting

When running the clock, do the Seconds displays increase 60 times faster than the Minutes displays?

- You may need to slow down the clock source, if it's been accelerated, in order to see the digits changing.

Use the following questions to help you troubleshoot and fix any issues with your circuit.

The seconds on the clock are changing too quickly.

- Try slowing down the clock source.

The seconds on the clock aren't increasing linearly from 0 to 59.

- Make sure the binary numbers are being converted to the segments of the display properly.

If you would like to add another extension to your clock, follow this link:

<https://mythinkscape.com/labs/v2/23200/steps/21146#Extensions>.

If you are done adding all the extensions you want, proceed to *1.7 Exercise: Final Testing*.

Setting Display Time

The following section discusses a function that allows you to manually set the display time. This section covers the research, planning, and construction of this additional feature.

To manually change the displayed clock time, you will need to overwrite the timing source that drives the counter in the clock. You will need a way to switch the timing source to something that you can manually control.

Answer the following questions to help guide you in your brainstorming and design process for the setting display time option:

1-16 Which counters do you want to be able to change manually: seconds, minutes, and/or hours? Consider the typical behavior of a real digital clock.

1-17 What kind of control on the Digital Electronics Board do you want to use to set the time? Consider using a button or switch.

Use the following guidelines to add the extension to the digital clock you built in the previous section, following the plans you just made.

- In the **Seconds**, **Minutes**, and/or **Hours** sub-circuits, you can use a logic circuit that will act as a switch between two input options (clock pulse and the manual set mode).

Testing and Troubleshooting

Manually set the time on your clock, then let it keep running. It should continue to count from the new time you've set.

- If you have made a 12-hour clock, ensure that you can set times in both AM and PM.

Use the following questions to help you troubleshoot and fix any issues with your circuit.

The minutes and hours on the clock aren't setting manually.

- If nothing is happening, make sure you are turning off the main pulse source before you try to enter the time manually.
- If it is increasing or decreasing inconsistently or incorrectly, check that your circuit is connected correctly and you are using the right kind of source.
- Make sure the manual time set does not let you go beyond 60 minutes and/or 24 (or 12) hours.
- Make sure that when you restart the time, the clock works properly.

If you would like to add another extension to your clock, follow the link here:
<https://mythinkscape.com/labs/v2/23200/steps/21146#Extensions>.

If you are done adding all the extensions you want, proceed to *1.7 Exercise: Final Testing*.

Alarm

The following section discusses an additional alarm you can add to the digital clock you've already built. This section covers the research, planning, and construction of this additional feature.

Answer the following questions to help guide you in your brainstorming and design process for the optional alarm clock:

1-18 To add an alarm clock you will need to create an additional circuit which will be unique to the alarm you decide to set. Therefore, every time you wish to change the alarm time you will need to create a new circuit. Decide what time you would like to set for your alarm. If you have a 12-hour clock, remember to include whether it is AM or PM. What time would you like to set for your alarm?

1-19 The alarm circuit needs to output a value of 1 only when the correct combination of numbers are displayed on the clock. Explain how you could use a logic converter to create this alarm logic circuit. Some numbers have unique truth table combinations that will provide a simpler logic circuit. See if you can use this to your advantage.

1-20 To keep your alarm going after the time changes, you will need to use a flip-flop. Describe how flip-flops work and which type you will need for this function of the alarm.

1-21 How would you add a turn off button to your circuit? What component should reset it?

1-22a What control on the Digital Electronics Board do you want to use as your turn off button to reset the alarm?

1-22b What PLD and FPGA connectors do you need for this control?

Note: Before you start building your circuit, sketch a block diagram detailing the alarm circuit and how it will be connected to the rest of your clock. Include the sketch with your finished lab.

Use the following guidelines to add to the digital clock you built in the previous section, following the plans you just made.

1. Add a sub-circuit named **Alarm** to your Clock PLD design.
2. Use the Logic Converter to build a truth table and then a circuit that generates a single high output when the clock reaches your alarm time.
3. Add in the flip-flop and counter to maintain the digital high that sets off the alarm after the time changes, and the clear/snooze button that turns the alarm off.
4. Add the appropriate PLD connector(s) for your alarm.
5. Wire the alarm to the Digital Electronics breadboard. Make sure you use the correct connectors on the board and ground your circuit.
 - o **WARNING:** make sure that the Digital Electronics Board is TURNED OFF before proceeding with this step. You should NEVER wire anything on the Digital Electronics Board while it's turned on.
6. Export the finished PLD design to the Digital Electronics Board.
7. Once you've built the new additions into your digital clock, take a screenshot, draw a sketch, or take a picture of the new sub-circuit and how it connects to the rest of the circuit, and a picture of your Digital Electronics Board (if you added hardware).

Note: Include the screenshot, sketch, or picture with your finished lab.

Testing and Troubleshooting

- Let your clock reach the time that sets off the alarm. Don't turn the alarm off until at least one minute has passed.

Use the following questions to help you troubleshoot and fix any issues with your circuit.

The alarm display isn't going off.

- Make sure you're using the right connectors in PLD and on the Digital Electronics Board.
- Make sure you've wired the alarm onto the breadboard correctly.

The alarm is going off at the wrong time.

- Make sure the circuit you make in the Logic Converter corresponds to the correct time.
 - Are you using the binary or seven-segment display outputs?
- Make sure you've connected the circuit from the Logic Converter to the counters properly.

The alarm turns off before the "snooze" button is pressed.

- Make sure the flip-flop is connected properly.

The "snooze" button doesn't turn off the alarm.

- Make sure you're clearing the flip-flop when you hit snooze.
- Make sure you're using the correct PLD and FPGA connectors and controls

If you would like to add another extension to your clock, follow the link here: <https://mythinkscape.com/labs/v2/23200/steps/21146#Extensions>.

If you are done adding all the extensions you want, proceed to *1.7 Exercise: Final Testing*.

Alarm with Buzzer or LED

The following section discusses an additional alarm buzzer or LED light you can add to the digital clock you've already built. You will be able to set the time of your alarm and have a buzzer or LED go off on the Digital Electronics breadboard. This section covers the research, planning, and construction of this additional feature.

Answer the following questions to help guide you in your brainstorming and design process for the optional alarm clock:

Note: if you have completed the *Alarm Extension*, skip to question 1-29.

1-23 To add an alarm to your clock you will need to create an additional circuit which will be unique to the alarm time you decide to set. Therefore, every time you wish to change the alarm time you will need to create a new circuit. Decide what time you would like to set your alarm. If you have a 12-hour clock, remember to include whether it's AM or PM.

1-24 The alarm circuit need to output a value of 1 only when the correct combination of numbers is displayed on the clock. Explain how you could use a logic converted to create this alarm logic circuit. Some numbers have unique truth table combinations that will provide a simpler logic circuit. See if you can use this to your advantage.

1-25 To keep your alarm going after the time changes you will need to use a flip-flop. Describe how flip-flops work and which type you will need for this function of the alarm.

1-26 How would you add a turn off button to your circuit? What component should it reset?

1-27 What control on the Digital Electronics Board do you want you use as your turn off button to reset the alarm? What PLD and FPGA connectors do you need for this control?

1-28 What do you want your alarm to be, an LED or a buzzer? If you do not have access to these pieces you can use the LED already on the Digital Electronics Board. Use the conditioning circuits below as a guideline.

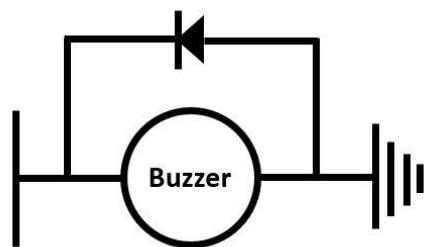


Figure 1-29 Buzzer conditioning circuit

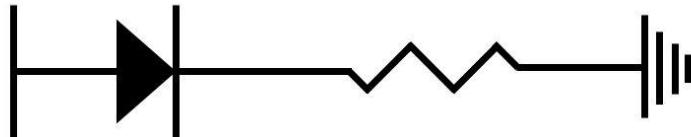


Figure 1-30 LED conditioning circuit

Note: Before you start building your circuit sketch a block diagram detailing the alarm circuit and how it will be connected to the rest of your clock. Include this sketch with your finished lab.

Use the following guidelines to add to the digital clock you built in the previous section, following the plans you just made.

1. Add a sub-circuit named **Alarm** to your Clock PLD design.
2. Use the Logic Converter to build a truth table and then a circuit that generates a single high output when the clock reaches your alarm time.
3. Add in the flip-flop and counter to maintain the digital high that sets off the alarm after the time changes, and the clear/snooze button that turns the alarm off.
4. Add the appropriate PLD connector(s) for your alarm.
5. Wire the alarm to the Digital Electronics breadboard. Make sure you use the correct connectors on the board and ground your circuit.
 - o **WARNING:** make sure that the Digital Electronics Board is TURNED OFF before proceeding with this step. You should NEVER wire anything on the Digital Electronics Board while it's turned on.
6. Export the finished PLD design to the Digital Electronics Board.
7. Once you've built the new additions into your digital clock, take a screenshot, draw a sketch, or take a picture of the new sub-circuit and how it connects to the rest of the circuit, and, do the same for your Digital Electronics Board (if you added hardware).

Note: Include your pictures, sketches, or screenshots with your finished lab.

Testing and Troubleshooting

Let your clock reach the time that sets off the alarm. Don't turn the alarm off until at least one minute has passed.

- Does the alarm turn on the LED or the buzzer?
- A digital output line may not provide enough power to turn on the device. You may need to use the digital signal to change the state of a relay or transistor, where an additional power supply provides power to the output device.

Use the following questions to help you troubleshoot and fix any issues with your circuit.

The alarm display isn't going off.

- Make sure you are using the right connectors in PLD and on the Digital Electronics Board.
- Make sure you have wired the alarm onto the breadboard correctly.

The alarm is going off at the wrong time.

- Make sure the circuit you make in the Logic Converter corresponds to the correct time.
 - o Are you using the binary or seven-segment display outputs?
- Make sure you have connected the circuit from the Logic Converter to the counters properly.

The alarm turns off before the "snooze" button is pressed.

- Make sure the flip-flop is connected properly.

The "snooze" button doesn't turn off the alarm.

- Make sure you are clearing the flip-flop when you hit snooze.
- Make sure you are using the correct PLD and FPGA connectors and controls

If you would like to add another extension to your clock, follow the link here:

<https://mythinkscape.com/labs/v2/23200/steps/21146#Extensions>.

If you are done adding all the extensions you want, proceed to *1.7 Exercise: Final Testing*.

1.7 Exercise: Final Testing

Instructions:

Take a video of your circuit as you go through the following tests:

- Let the clock run at the regular speed for 2 minutes.
- Let the clock run through a full accelerated 24-hour cycle.
- **12-hour clock:** Show the time on the clock transitioning from 12 (AM or PM) to 1 (AM or PM).
- **12-hour clock:** Show the AM/PM indicator changing between the 11th and 12th hours.
- Optional Extensions:
 - **Alarm and Alarm buzzer LED:** Show that the alarm goes off at a specific time, keeps going for more than a minute, and can be turned off using the "snooze" button.
 - **Setting Display Time:** Manually set the time, then let the clock continue running.
 - **Seconds Display:** Show that the clock displays 6 digits; HH:MM:SS.

Note: Include your video with your finished lab.

Alternatively, if you are unable to record video, record your observations below:

1.8 Conclusion

1-29 Describe the difference between an analog and a digital clock. List the necessary requirements for both clocks to function and two different applications for each (not mentioned in this lab).

1-30 What is the role of the clock pulse in this circuit? What characteristics of the clock pulse did you change to speed up your clock for testing?

1-31 Explain how the counters roll over and clear when a certain number is attained.

1-32 How is the decoder used in this lab?

1-33 Did you build an alarm for this clock?

- **No:** How would you set the alarm time in software? (See section 1.6: *Extensions* for hints)
- **Yes:** How would you set the alarm time in hardware?

1-34 How would building this circuit in Multisim instead of PLD change your design?

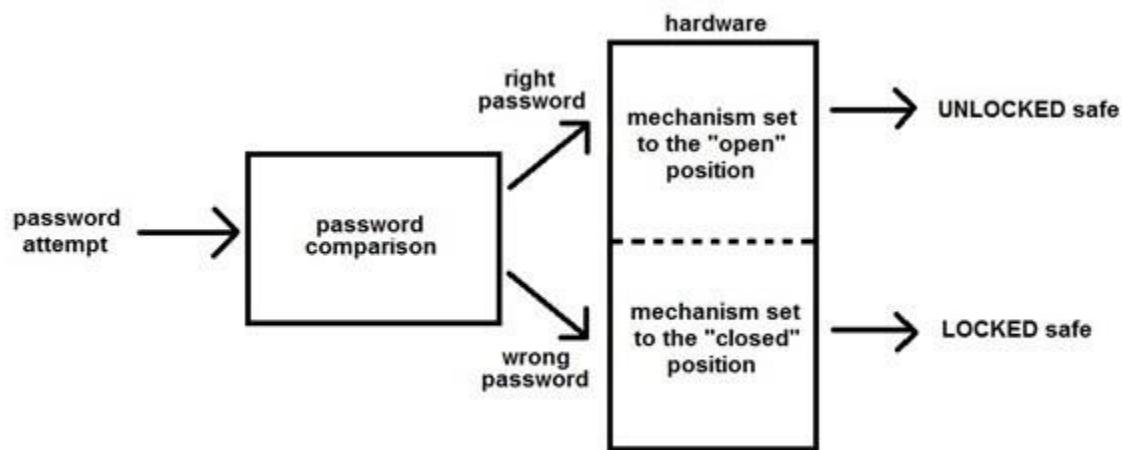
1-35 What was one unexpected challenge you faced during this project? How did you overcome it?

1-36 If you could go back and re-build the circuit, what changes would you make? Would you approach the project differently?

Lab Manual:

Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Lab 17: Electronic Safe – Application Lab #2

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 17: Electronic Safe – Application Lab #2

The keypad converts each digit entered to a 4-bit binary number. To store a total of 16 bits, four serial-to-parallel shift registers will be used. Each shift register will be built to store the 4 most recent bits of data entered. The first shift register will store the first bit from each digit, the second shift register will store the second bit from each digit, etc.

The output of the shift registers will then be compared to the correct password (input using digital highs and low) using XNOR gates. When the password attempt is correct, the combination of XNOR and AND gates will output a 1 when all of the digits of the password are correct.

At the end of your project, you should have an electronic safe simulation on the Digital Electronics Board. It will have a 4-digit password that is set using constants in the PLD design. Password attempts will be made using the four buttons on the board. If the correct password is entered, a green LED will light up.

Additionally, there are extensions you can build into your safe. These include tracking the number of incorrect password attempts and setting off an alarm if too many are made, a solenoid to simulate physically opening the safe door, and the ability to change the password.

Before starting this lab, it is suggested that you review the following labs from previous weeks:

- Lab 4 - Logic gates Explored and Boolean Algebra
- Lab 10 - Flip-flops
- Lab 13 - Shift Registers
- [Optional]: Lab 11 - Counters

Learning Objectives

In this lab, students will:

1. Build an electronic safe with a 4-digit password.
2. Explore how flip-flops can be combined to make shift registers.
3. Explore the practical applications of combinational logic circuits.
4. Have the option to learn how to wire additional hardware to the Digital Electronics breadboard.

Required Tools and Technology

Platform: NI ELVIS III

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>

Hardware: [Optional] Electronic Components

Buzzer/LED Password Attempts:

- (1) Red LED
- (1) 1000 Ω resistor
- Wire

OR

- (1) 5 V buzzer
- (1) IN3064 Diode
- Wire

Solenoid Hardware Addition:

- (1) 5 V Solenoid

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM/US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Software: NI LabVIEW FPGA Vivado 2014.4

- ✓ Install:
<http://www.ni.com/download/labview-fpga-module-2015->

sp1/5920/en/

Note: Digilent Driver (The installer above automatically downloads the installer below onto your computer)

- ✓ Navigate to:
C:\NIFPGA\programs\Vivado2
014_4\data\xicom\cable_driver
s\nt64\digilent
- ✓ Install: install_digilent.exe

Expected Deliverables

In this lab, you will collect the following deliverables:

- Screenshot, Picture, or Sketch of circuit and sub-circuits
- Screenshots, Pictures, or Sketches of any changes you make to your circuit
- [Optional] Taking incorrect password attempts extension drawings and screenshots
- [Optional] Buzzer or LED password attempts extension screenshots, pictures, or sketches
- [Optional] Solenoid extension drawings, screenshots, or pictures
- [Optional] Setting a new password extension drawings, screenshots, or pictures
- Circuit video
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

Electronic Safe

Introduction Video



Figure 1-1 Video View the video here: <https://youtu.be/DtsCrieWKk4>



Video Summary

- This video will explore how many of the concepts and skills you have learned in this course will help you design and build an electronic safe
- Digital clocks are very commonly used in hotels and homes to hold small, valuable objects
- Electronic safes use a logic circuit to compare password attempts to the correct password
- Hardware opens an electronic safe when the correct password is entered

Electronic Safes

Electronic safes are composed of a logic circuit which compares password attempts to the correct password, and hardware that opens the safe's locking mechanism when the correct password is entered. The password is typically 4-digits long and entered using a keypad on the main panel of the safe.

More specifically, the electronic safe you will create in this lab contains a series of shift registers to store password data, a logic circuit that compares the stored password data to the correct password, and a hardware component to control the safe's locking mechanism. The password will be 4 numeric digits, entered using a keypad. When the correct password is entered, a green LED will turn on to indicate that the safe is "open."

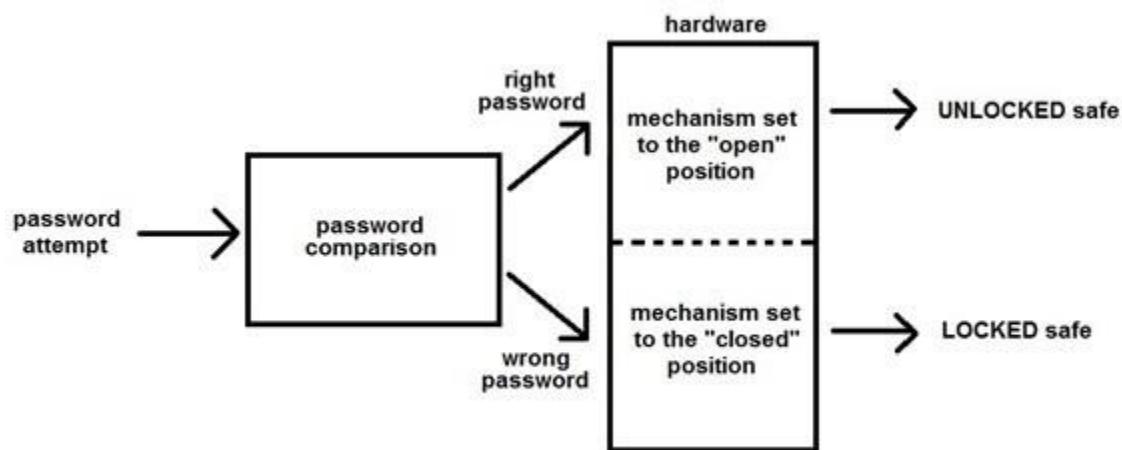
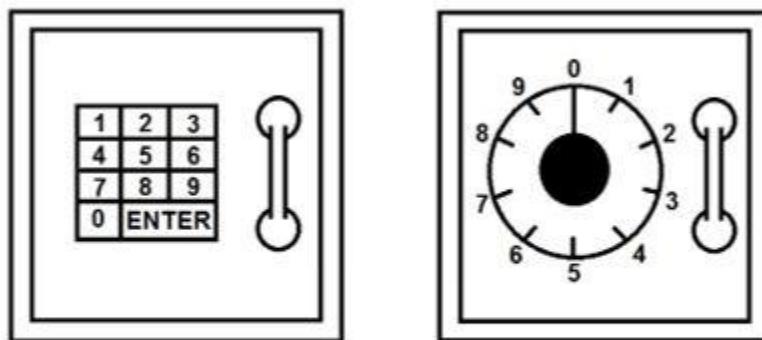


Figure 1-2 Safe diagram

Traditional *combinational safes* work differently from electronic safes because they are controlled solely by hardware. Combination safes are opened by a dial found on the front panel of the safe. The passcode consists of a series of numbers or letters set in a specific sequence, which are shown on the face of the dial.



Left: *Electronic Safe Keypad* Right: *Combination Safe Dial*

Figure 1-3 Combination safe

Electronic safes are commonly used to secure small valuables in hotel rooms and everyday households. One advantage available with most electronic safes is that the owner can set several different combinations to be correct, meaning that multiple combinations will be able to open the safe. This is especially convenient when several people need to have access to a safe, like in a workplace. Each employee can have their own password, which can be activated or deactivated as needed without affecting the other passwords. The safe can also be programmed to track when and how often each password is used. Since we're building a simple electronic safe in this lab, it will not possess these features.

Research and Planning

- Research and summarize the following keywords to help you gain a better understanding of what you will need in this lab.
- You will be able to come back and add more information at any point in the lab.

1-1 Electronic safe definition:

1-2 Digital lock definition:

1-3 Flip-flops definition (what is available in Multisim):

1-4 Serial-in-parallel-out (SIPO) shift registers definition:

1-5 Breadboards definition (how do you build circuits on them?) :

Answer the following questions to help guide you through your brainstorming and design process.

1-6 Make a checklist of all of the different things your circuit should be able to accomplish.

1-7 What logic gate will you need to convert digits inputted with the buttons on the Digital Electronics Board to binary digits?

1-8 How many shift registers will there be for a 4-digit decimal password? How many flip-flops will you need for each shift register?

1-9 How do you set a correct password? Remember that interactive digital constants do not work in PLD designs.

1-10 How do you compare the correct password to the entered password? Remember that you will need one kind of logic gate to compare the individual bits and output a 1 when they match, and another kind of logic gate to output a single 1 when all of the comparisons output a 1.

1.2 Exercise: Building the Electronic Safe Components

Use the following guidelines to build your electronic safe. A block diagram of the overall safe design is show below:

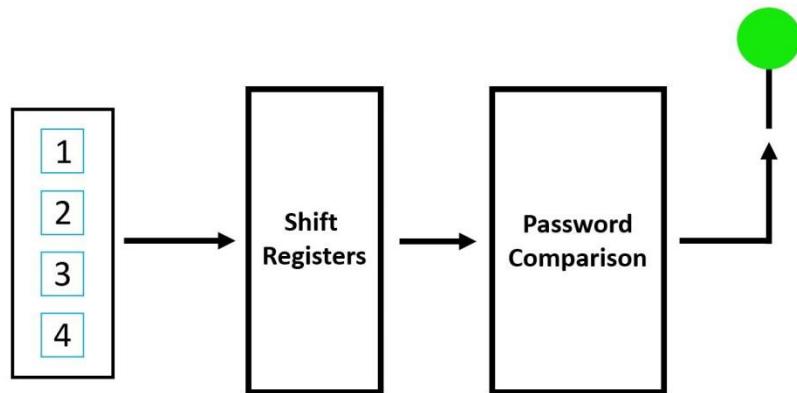


Figure 1-4 Electronic safe guidelines

Tip: Add both FPGA controls and interactive digital constants, as well as probes, so that you can test the different components of your circuit without having to wait for it to export to the Digital Electronics Board.

1. Open a **NEW** PLD file.
2. Place three PLD sub-circuits in the main PLD workspace, and name them **Keypad**, **Shift Registers**, and **Password Comparison**.

Components Overview

- Keypad Sub-circuit:
<https://mythinkscape.com/labs/v2/23293/steps/21191#Keypad%20SubC>
- Shift Registers Sub-circuit:
<https://mythinkscape.com/labs/v2/23293/steps/21191#Shift%20Registers%20SubC>
- Password Comparison Sub-circuit:
<https://mythinkscape.com/labs/v2/23293/steps/21191#Password%20Comparison%20SubC>

Keypad Sub-circuit

1. In the **Keypad** sub-circuit, place the following components:
 - (2) OR2 gates and (1) OR4 gate
 - (1) digital low
2. Add (4) PLD input connectors and name them **Key1 – Key4**.
3. Add (5) PLD output connectors and name them:
 - **Bit1 – Bit4**
 - **Enter**
4. Once all of the components have been placed in the workspace, wire the Keypad circuit as shown in the diagram below:

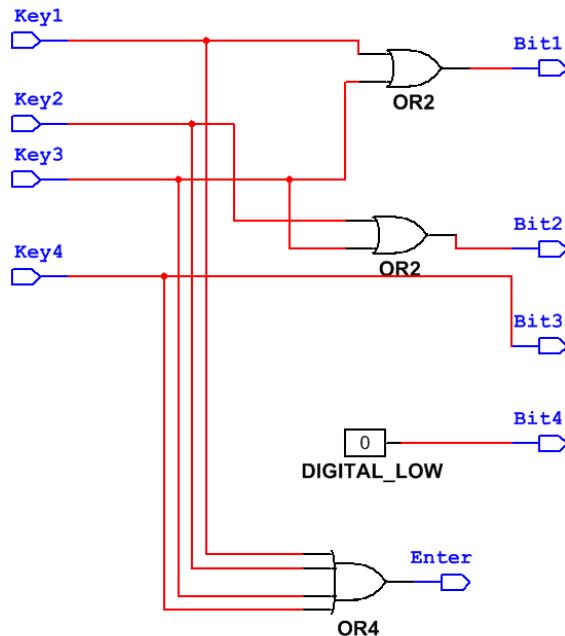


Figure 1-5 Keypad sub-circuit

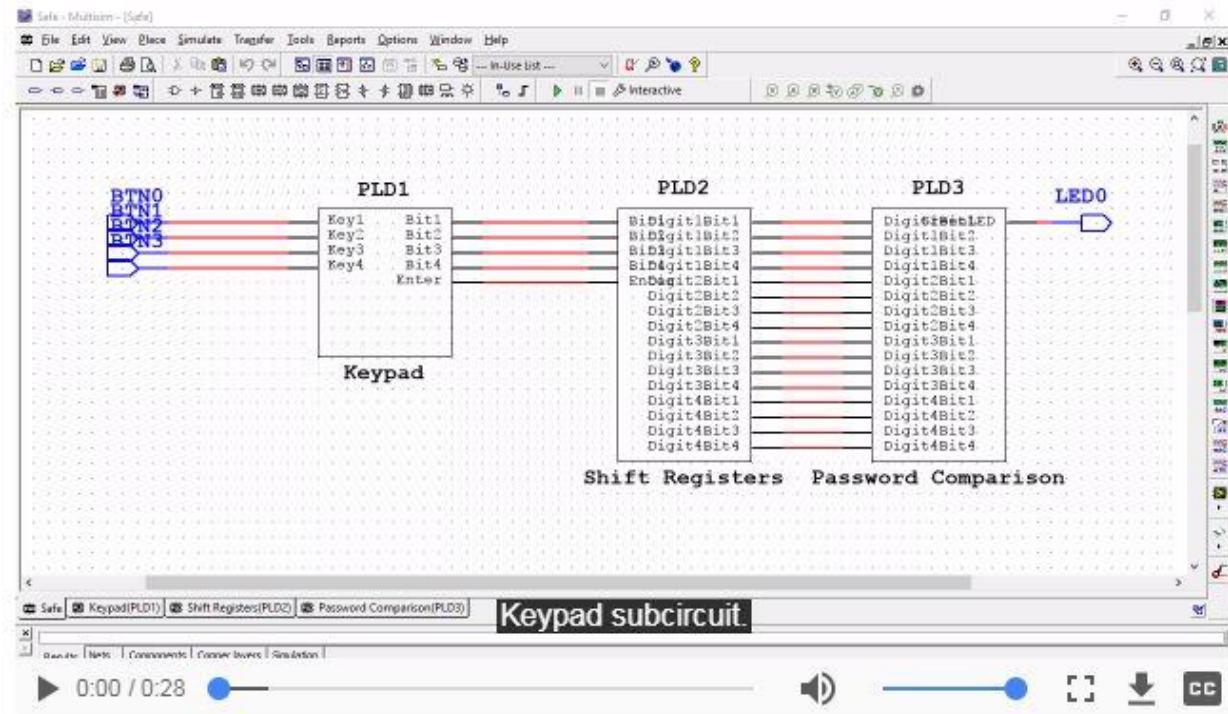


Figure 1-6 Keypad sub-circuit video View the video here: https://cf-ts.mythinkscape.com/video/Keypad_no_cursor.mp4

Shift Registers Sub-circuit

1. Open the sub-circuit **Shift Registers** and place the following components:
 - (16) D-Flip-Flops (FF_D_PCLR_CO)

Note: These flip-flops build the shift registers that hold the password attempt data so it can be compared to the set password.

- (1) digital high

Note: This runs and powers the flip-flops.

- (2) digital high
2. Add (5) PLD input connectors and name them:
 - **Bit1** to **Bit4**
 - **Enter**
 3. Add (16) PLD output connectors and name them:
 - **Digit1Bit1** to **Digit1Bit4**
 - **Digit2Bit1** to **Digit2Bit4**
 - **Digit3Bit1** to **Digit3Bit4**
 - **Digit4Bit1** to **Digit4Bit4**

4. Once all of the components have been placed in the workspace, wire the Shift Registers circuit as shown in the diagram below:

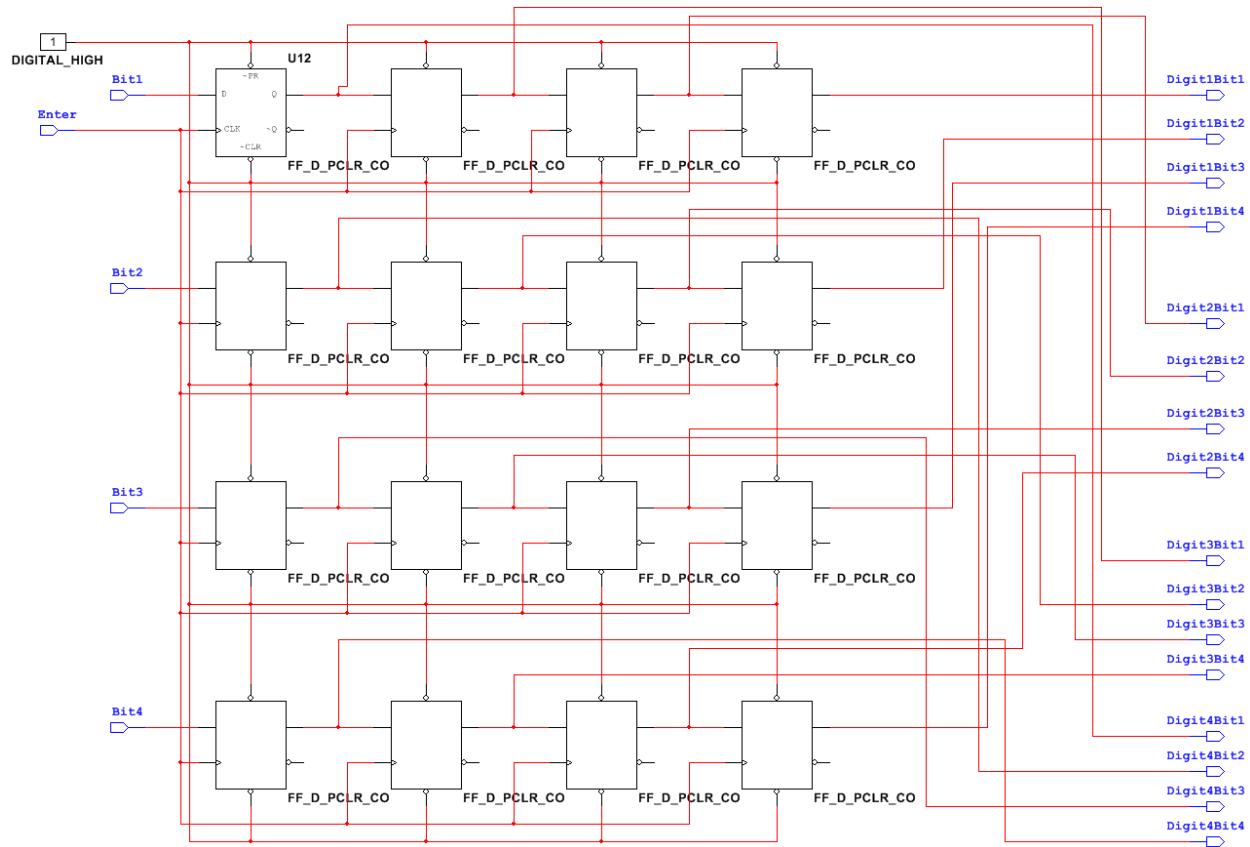


Figure 1-7 Shift registers sub-circuit

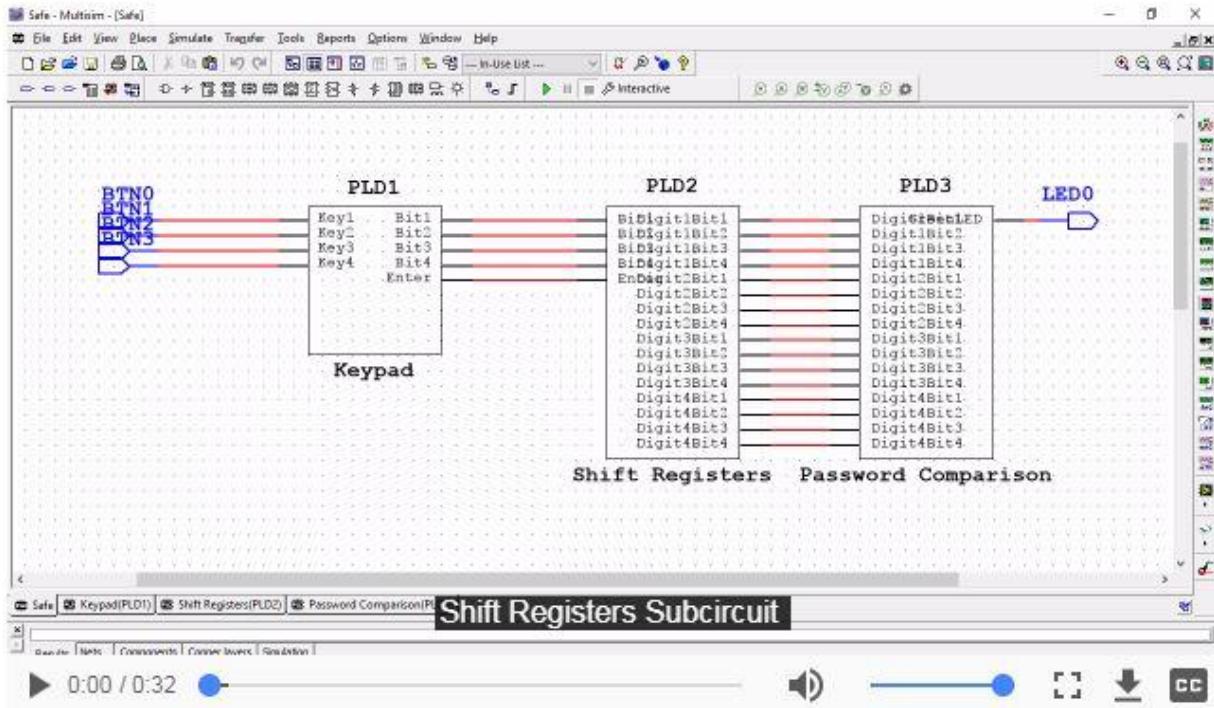


Figure 1-8 Shift registers sub-circuit video View the video here: https://cts.mythinkscape.com/video/Shift_Registers_no_cursor.mp4

Password Comparison Sub-circuit

1. In the **Password Comparison** sub-circuit place the following components:

Note: Choose your digital highs and lows to set the correct password. Make sure to place the highs and lows in the correct order. For example, to enter the number 1 (binary 0001) **Digit1Bit1** would be 1, **Digit1Bit2** would be 0, **Digit1Bit3** would be 0, and **Digit1Bit4** would be 0.

- (16) XNOR2 gates

Note: These gates compare the attempted password to the set password and output a high when the two match.

- (16) digital lows and highs

Note: These allow you to set the correct password.

- (2) AND8 gates

Note: These gather the outputs from the XNOR gates and output a high when the correct password has been entered.

- (1) AND2 gate

Note: Connects the outputs of the XNOR outputs to the green LED.

2. Add (16) PLD input connectors and name them:
 - **Digit1Bit1** to **Digit1Bit4**
 - **Digit2Bit1** to **Digit2Bit4**
 - **Digit3Bit1** to **Digit3Bit4**
 - **Digit4Bit1** to **Digit4Bit4**
3. Add (1) PLD output connectors and name it **GreenLED**.
 - **HourSSD1a** to **HourSSD1g**
 - **HourSSD2a** to **HourSSD2g**
4. Once all the components have been placed in the workspace, wire the Hours sub-circuit as shown in the diagram below.

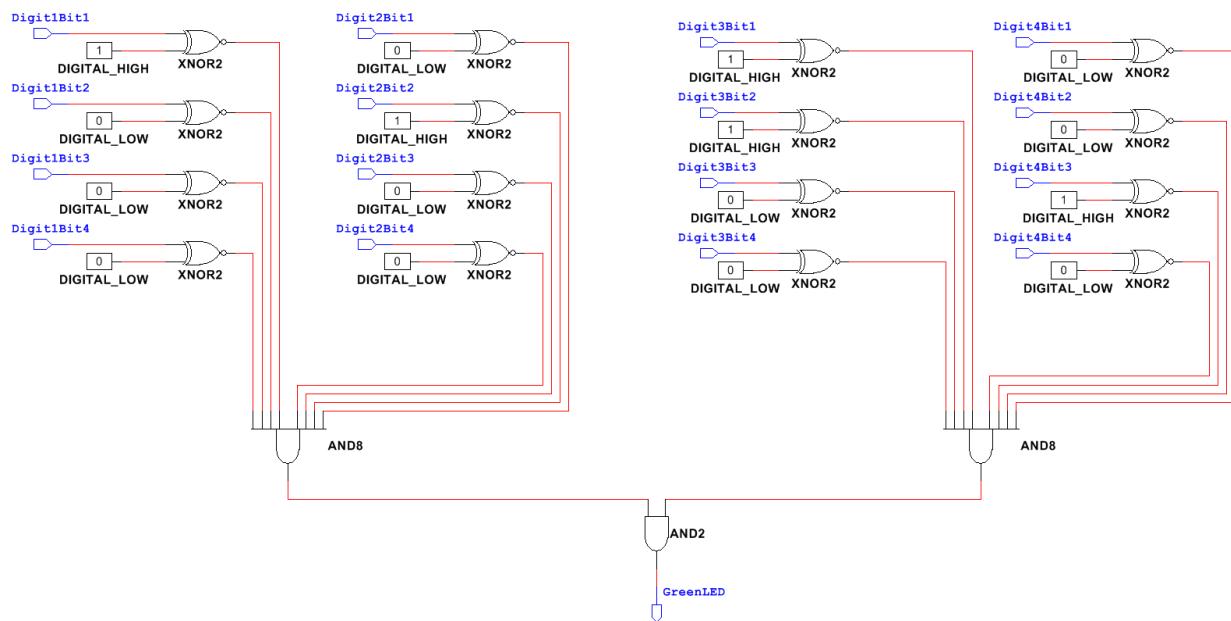


Figure 1-9 Password comparison sub-circuit

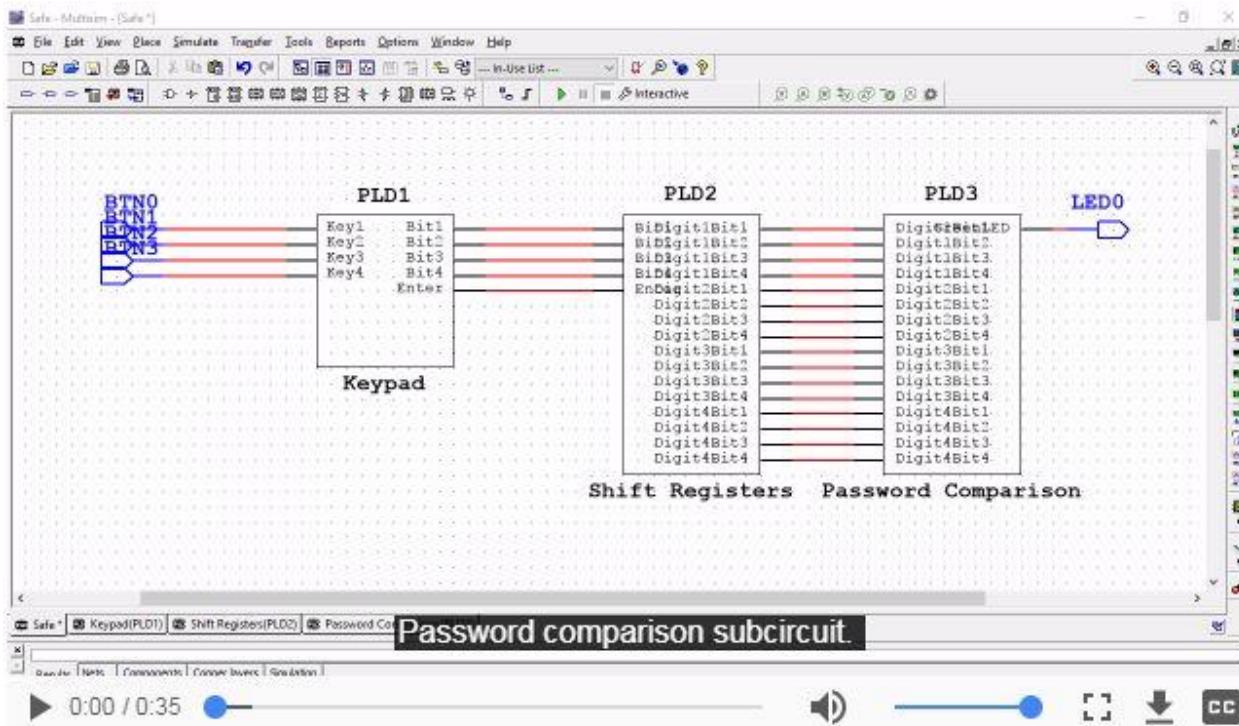


Figure 1-10 Password comparison sub-circuit video View the video here: https://cf-ts.mythinkspase.com/video/Password_Comparision_no_cursor.mp4

1.3 Exercise: Putting It All Together

Instructions:

1. Place (4) PLD input connectors for the buttons **BTN0** to **BTN3** in the main PLD design.
2. Place the PLD output connector **LEDO** in the main PLD design.
3. Connect all of the sub-circuits as show in the figure below to complete your electronic safe.

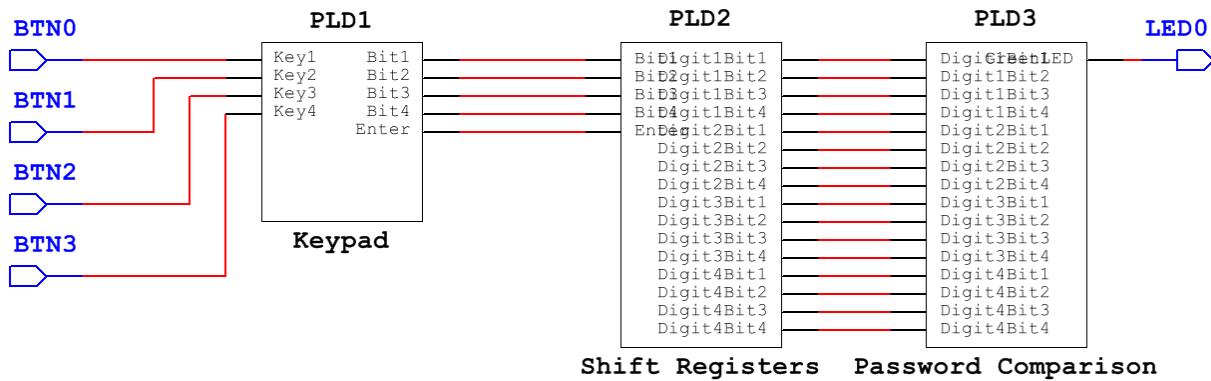


Figure 1-11 Digital Circuit

4. Deploy the circuit to the Digital Electronics Board.

Note: Take a screenshot, draw a picture, take a photo of your finished circuit in PLD as well as the individual sub-circuits and include them with your finished lab.

1.4 Testing and Troubleshooting

Go through the following steps to test that your safe is working properly. If you come across any issues, use the questions later in this section to help you troubleshoot and fix mistakes.

Instructions:

1. Export your PLD circuit to the Digital Electronics Board. Make sure you restart the board and it is turned on before you export your circuit.
2. Enter the correct password.
 - Does the **green LED** turn on?
3. Enter an incorrect password.
 - Does the **green LED** turn off?

Use the following questions to help you troubleshoot and fix any issues with your circuit:

Things to always check:

- Did you turn your Digital Electronics Board off and on before deploying it?
 - The board needs to be turned off and on to reset

The PLD circuit will not deploy to the Digital Electronics Board.

- Ensure you have turned on the NI ELVIS III and the Digital Electronics Board.
- Ensure that the board is connected to your computer and to power. Sometimes the USB port on your computer isn't making a good connection, so try using a different one.
- Be sure to select the correct board when prompted.
- Be sure to select the correct Xilinx tool (either 32-bit or 64-bit depending on your operating system).
- Check that the correct connectors have been chosen in PLD. Their names must match up with those on the Digital Electronics Board. Also, make sure that they are correctly wired in PLD and on the board.

When the correct password is entered, the LED does not turn on.

- Add a probe to your PLD design and simulate the circuit to check if the issue is with the simulated circuit or the hardware circuit.

PLD design issues:

Are you using the keypad correctly?

- Make sure that the keypad is outputting the correct binary numbers.
- Make sure that the binary output of the keypad is correctly aligned with the input of the shift registers. For example, in this lab, 1 (binary 0001) would be divided as **Bit1 = 1**, **Bit2 = 0**, **Bit3 = 0**, and **Bit4 = 0**.

Are the shift registers wired correctly?

- Make sure that the **Enter** button is wired to the **CLK** pin of *EVERY* flip-flop.
- Make sure that the **digital high** is wired to the **~PR** and **~CLR** pins of *EVERY* flip-flop.
- Make sure that the flip-flops are connected together in series by their **D** and **Q** pins.

Are you comparing the shift register outputs with their corresponding correct password bits?

Use probes to see the output of the shift registers directly. Do you have the shift register outputs connected to the proper bits of the correct password? Use the example below as a guide. In the example, each square represents a flip-flop, arranged in the same way that they are in the figures earlier. The password attempt that has been entered is “1234”.

0	1	0	1
0	1	1	0
1	0	0	0
0	0	0	0

Figure 1-12 Password attempt

Do you have the correct password set properly?

- Are you using digital highs and digital lows? Interactive digital constants do not work on the Digital Electronics Board.
- Have you converted from decimal to binary numbers correctly? Use the table below to verify.

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Figure 1-13 Decimal to binary conversion table

Note: If you had to make any fixes, take a screenshot, draw a sketch, or take a photo of your completed circuit in PLD and include it with your finished lab.

1.5 Extensions

The following section discusses different additions you can make to the electronic safe you have already built. Choose *ONE or more* of these extensions to add to your safe:

- Tracking incorrect password attempts:
<https://mythinkscape.com/labs/v2/23293/steps/21195#Tracking%20Incorrect%20Password%20Attempts>
- Adding a buzzer or LED that turns on when too many wrong password attempts are made: <https://mythinkscape.com/labs/v2/23293/steps/21195#Buzzer/LED>
- Adding a solenoid that releases when the correct password is entered (this simulates safe unlocking):
<https://mythinkscape.com/labs/v2/23293/steps/21195#Solenoid>
- Setting a new password:
<https://mythinkscape.com/labs/v2/23293/steps/21195#Setting%20a%20New%20PW>

Tracking Incorrect Password Attempts

The following section discusses an additional feature you can add to the safe you have already built. This feature allows the safe to track the number of incorrect password attempts. This section covers the research, planning, and construction of this additional feature.

Research and summarize the following keywords to help you gain a better understanding of what you will need in this part of the lab. You will be able to come back and add more information at any point in the lab.

1-11 Digital counters (output type, clear and clock functions, etc.):

1-12 Actuators:

Answer the following questions to help guide you through your brainstorming and design process. These questions will also ensure your research covered the necessary information for you to expand on your safe.

1-13 How many incorrect attempts do you want to allow before the alarm goes off?

1-14 What kind of counter do you want to use?

1-15 How do you track incorrect password entries?

1-15a How would you track all password entries?

1-15b What output indicated that a password entry was wrong?

1-15c How would you combine these two things so that adding to the count is depending on them both being true?

1-16 How do you reset the counter every time the correct password is entered?

Make detailed drawings of each piece of the circuit you plan to add to your safe. In addition, sketch how you plan to insert these additions into the pre-existing circuit.

Note: Include the drawings with your finished lab.

Use the following guidelines to add to the password-controlled safe you built in the previous section, following the plans you just made.

1. Add the counter that tracks the number of incorrect password entries made. Make sure the counter clears when a correct password entry is made. Test your circuit to make sure the counter is working properly before moving on.
2. Add the flip-flop to the counter so the alarm (probe) will keep going off when more incorrect password entries are made. Test your circuit to make sure the flip-flop is working properly before moving on.
3. Add the connector for the probe to your simulated circuit.

Note: Take a screenshot, sketch, or picture of the additions to your PLD design and include it with your finished lab.

Testing and Troubleshooting

1. Enter enough incorrect passwords to set off the alarm (probe).
 - Does the alarm go off?
 - Enter another incorrect password. Is the alarm still going off?
 - Enter the correct password. Does the alarm turn off?

Use the following questions to help you troubleshoot and fix any issues with your circuit.

The alarm going is off too late/soon or not at all.

- In general, use probes and interactive digital constants to test this particular portion of the circuit independently.
- Is the counter wired correctly?
 - Are you using the correct counter output? Remember the output is usually in binary numbers. Use *Figure 1-11 Decimal to binary conversion table* to help you with this step.

- Does the counter clear when the correct password is entered?
What output in the main circuit would you use to clear the counter?
Is the clear pin positive or negative edge triggered?
 - Is your counter positive or negative edge triggered? Which one do you want for your circuit?
 - What is your counter counting? Choose something that changes every time a new digit is entered.
 - Does your counter stop counting once it has reached the maximum number of attempts? If not, consider making the input to the counter dependent on the output of the counter being below a number.

- Is the flip-flop working properly?
 - Test the behavior of your flip-flop using interactive digital constants and probes.
 - Is the output of the flip-flop 1 when the counter outputs a 1?
 - Does the flip-flop maintain an output of 1 after the counter continues to go up?
 - Does the flip-flop return to 0 when the correct password is entered?

Note: If you had to make any fixes, take a screenshot, draw a sketch, or take a photo of your updated PLD and include it with your finished lab.

If you would like to add another extension to your clock, follow the link here: <https://mythinkscape.com/labs/v2/23293/steps/21195#Extensions>.

If you are done adding all the extensions you want, proceed to *1.7 Exercise: Final Testing*.

Buzzer or LED Password Attempts

The following section discusses an additional alarm (red LED or buzzer) you can add to the safe you've already built. The alarm sets off if too many wrong password attempts are made. This section covers the research, planning, and construction of this additional feature.

Research and summarize the following keywords to help you gain a better understanding of what you will need in this part of the lab. You will be able to come back and add more information at any point in the lab.

Note: if you have already added a *Tracking Incorrect Password Attempts* extension to your safe, skip to question 1-19.

1-17 Digital counters (output type, clear and clock functions, etc.):

1-18 Actuators:

1-19 Buzzers: What do they do?

1-20 Buzzers: How do they work?

1-21 Buzzers: What are the specifications for the buzzer you will be using?

1-22 Buzzers: How do you wire a buzzer?

1-23 LEDs (the basic structure, how to connect them to power, etc.):

Answer the following questions to help guide you through your brainstorming and design process. These questions will also ensure your research covered the necessary information for you to expand on your safe.

Note: If you have already added a *Tracking Incorrect Password Attempts* extension to your safe, skip to question 1-28.

1-24 How many incorrect attempts do you want to allow before the alarm goes off?

1-25 What kind of counter do you want to use?

1-26 How do you track incorrect password entries?

1-26a How would you track all password entries?

1-26b What output indicates that a password entry was wrong?

1-26c How would you combine these two things so that adding to the count is dependent on them both being true?

1-27 How do you reset the counter every time the correct password is entered?

1-28 How would you connect the counter so that it sets off the alarm (LED or buzzer)?

1-29 How do you keep the alarm going off when more incorrect passwords are entered? Could a flip-flop be useful?

1-30 How do you turn off the alarm after the correct password is entered?

Adding a buzzer or LED:

1-31 What do you want the final circuit to do?

1-32 Which output on your pre-existing circuit can you use to control the buzzer/LED?

1-33 What does the circuit connecting the buzzer on the breadboard look like?

Make detailed drawings of each piece of the circuit you plan to add to your safe. In addition, sketch how you plan to insert these additions into the pre-existing circuit. Include these with your finished lab.

Use the following guidelines to add to the password-controlled safe you built in the previous section, following the plans you just made.

Note: If you have already added a *Tracking Incorrect Password Attempts* extension to your safe, skip to step 3.

Counting Incorrect Password Entries:

1. Add the counter that tracks the number of incorrect password entries made. Make sure the counter clears when a correct password entry is made. Test your circuit to make sure the counter is working properly before moving on.
2. Add the flip-flop to the counter so the alarm will keep going off when more incorrect password entries are made. Test your circuit to make sure the flip-flop is working properly before moving on.
3. Add the connector for the alarm to your simulated circuit.
 - **WARNING:** make sure that the Digital Electronics Board is TURNED OFF before proceeding with this step. You should NEVER wire anything on the Digital Electronics Board while it's turned on.
4. Add your alarm to the breadboard on the Digital Electronics Board using the circuit you created in your design plan. Make sure you use the correct connectors on the board and ground your circuit.

Adding a Buzzer or LED:

1. Use the diagrams below to help you wire the buzzer or LED
 - **WARNING:** make sure that the Digital Electronics Board is TURNED OFF before proceeding with this step. You should NEVER wire anything on the Digital Electronics Board while it's turned on.

2. Add your buzzer to the breadboard on the Digital Electronics Board using the circuit you created in your design plan. Make sure you use the correct connectors on the board and ground your circuit.

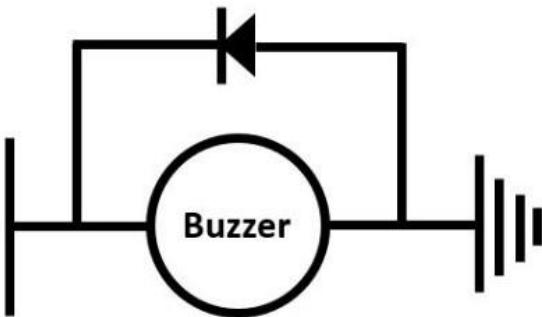


Figure 1-14 Buzzer

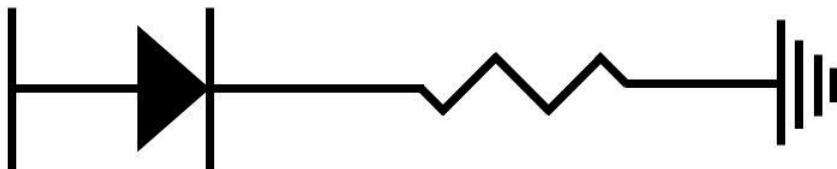


Figure 1-15 LED

Note: Take a screenshot of all the additions you make to your PLD design and a picture of any hardware additions you make, and upload it to the Notes section.

Testing and Troubleshooting

1. Enter the correct password.
 - Does the green LED turn on?
2. Enter an incorrect password.
 - Does the green LED turn off?
3. Enter enough incorrect passwords to set off the alarm.
 - Does the alarm go off?
 - Enter another incorrect password. Is the alarm still going off?
 - Enter the correct password. Does the alarm turn off?

Use the following questions to help you troubleshoot and fix any issues with your circuit:

The alarm is going off too late/soon or not at all.

- In general, use probes and interactive digital constants to test this particular portion of the circuit independently.
- Is the counter wired correctly?
 - Are you using the correct counter output? Remember the output is usually in binary numbers. Use *Figure 11 - Decimal to Binary Converter* to help you.
 - Does the counter clear when the correct password is entered? What output in the main circuit could you use to clear the counter? Is the clear pin positive or negative edge triggered?
 - Is your counter positive or negative edge triggered? Which one do you want for your circuit?
 - What is your counter counting? Choose something that changes every time a new digit is entered.
 - Does your counter stop counting once it's reached the maximum number of attempts? If not, consider making the input to the counter dependent on the output of the counter being below a certain number.
- Is the flip-flop working properly?
 - Test the behavior of your flip-flop using interactive digital constants and probes.
 - Is the output of the flip-flop 1 when the counter outputs a 1?
 - Does the flip-flop maintain an output of 1 after the counter continues to go up?
 - Does the flip-flop return to 0 when the correct password is entered?
- Is the buzzer/LED correctly?
 - Are you using the same connector on the breadboard as in the PLD design?
 - Is the anode connected to ground and cathode connected to power?
 - Is the frequency of your buzzer set within the audible range?

Note: If you had to make any fixes, take a screenshot, draw a picture, or take a photo of your updated circuit in PLD and on the Digital Electronics Board, and include it with your finished lab.

If you would like to add another extension to your safe, follow the link here:
<https://mythinkscape.com/labs/v2/23293/steps/21195#Extensions> .

If you are done adding all the extensions you want, proceed to *Step 7 - Exercise: Final Testing*.

Solenoid

The following section discusses an additional solenoid you can add to the safe you've already built. The solenoid releases when the correct password is entered (this simulates the safe unlocking). This section covers the research, planning, and construction of this additional feature.

Research and summarize the following keywords to help you gain a better understanding of what you will need in this part of the lab. You will be able to come back and add more information at any point in the lab.

1-34 What do solenoids do?

1-35 How do solenoids work?

1-36 What are the specifications for the solenoid you will be using?

1-37 How do you wire a solenoid?

Answer the following questions to help guide you through your brainstorming and design process. These questions will also ensure your research covered the necessary information for you to expand on your safe.

1-38 What do you want the final circuit to do?

1-39 Which output on your pre-existing circuit can you use to control the solenoid?

1-40 What does the circuit connecting the solenoid on the breadboard look like?

Use the following guidelines to add to the password-controlled safe you built in the previous section, following the plans you just made.

1. Use the diagram below to help you wire the solenoid.
 - o **WARNING:** make sure that the Digital Electronics Board is TURNED OFF before proceeding with this step. You should NEVER wire anything on the Digital Electronics Board while it's turned on.
2. Add your solenoid to the breadboard on the Digital Electronics Board using the circuit you created in your design plan. Make sure you use the correct connectors on the board and ground your circuit.



Figure 1-16 Solenoid

Note: Take a screenshot, draw a picture, or take a photo of all the additions you make to your PLD design and a picture of any hardware additions you make, and include them with your finished lab.

Testing and Troubleshooting

1. Enter the correct password.
 - o Does the green LED turn on?
 - o Does the solenoid open?

Use the following questions to help you troubleshoot and fix any issues with your circuit:

The solenoid isn't moving at all:

- Is the solenoid wired correctly on the Digital Electronics Breadboard?
- Do you have the correct FPGA and PLD connectors?
- Do you have power and ground connected?
- Do you have the right voltage passing through your solenoid?

The solenoid is moving in the wrong direction:

- Do you have the correct FPGA and PLD connectors?
- Try adding an inverter to switch the output to the solenoid from high to low or vice versa.

Note: If you had to make any fixes, take a screenshot, draw a sketch, or take a photo of your completed circuit in PLD and on the Digital Electronics Board, and include it with your finished lab.

If you would like to add another extension to your safe, follow this link:
<https://mythinkscape.com/labs/v2/23293/steps/21195#Extensions> .

If you are done adding all the extensions you want, proceed to *Step 7 - Exercise: Final Testing*.

Setting A New Password

The following section discusses an additional feature you can add to the safe you've already built. This feature allows you to reset the password for the safe. This section covers the research, planning, and construction of this additional feature.

Answer the following questions to help guide you through your brainstorming and design process. These questions will also ensure your research covered the necessary information for you to expand on your safe.

1-41 How would you clear the original password?

1-42 How do you store a new password?

1-43 How do you turn on/off the “reset” mode?

Make detailed drawings of each piece of the circuit you plan to add to your safe. In addition, sketch how you plan to insert these additions into the pre-existing circuit.

Note: Upload these drawings to the Notes section.

Use the following guidelines to add to the password-controlled safe you built in the previous section, following the plans you just made.

1. Add components needed in the circuit to turn on/off the "reset" mode.
2. Make changes to the Password Comparison sub-circuit to allow the password to be modified.

Note: Take a screenshot of all the additions you make to your PLD design and a picture of any hardware additions you make, and upload it to the Notes section.

Testing and Troubleshooting

1. Enter the correct password.
 - o Does the green LED turn on?
2. Change the password, and then enter the new password.
 - o Does the green LED turn on?

Use the following questions to help you troubleshoot and fix any issues with your circuit:

New password does not open the safe.

- In general, use probes and interactive digital constants to test this particular portion of the circuit independently.
- Are the components wired correctly?
- Is the new password stored successfully?
 - o Is the Password Comparison sub-circuit comparing the set password and the input digits?

Note: If you had to make any fixes, take a screenshot, draw a sketch, or take a photo of your completed circuit in PLD and on the Digital Electronics Board, and include it with your finished lab.

If you would like to add another extension to your safe, follow the link here:
<https://mythinkscape.com/labs/v2/23293/steps/21195#Extensions> .

If you are done adding all the extensions you want, proceed to *Step 7 - Exercise: Final Testing*.

1.6 Exercise: Final Testing

Instructions:

Take a video of your circuit as you go through the following tests:

- What is the correct password (the one that "opens" your safe)?
- Enter the correct password.
 - Does the green LED turn on?
 - **Optional:** does the solenoid "open"?
- Enter an incorrect password.
 - Does the green LED turn off?
- **Optional:** enter enough incorrect password to set off your alarm.
 - Does the alarm go off?
 - Does the alarm keep going off if you enter an incorrect password again?
 - Does the alarm turn off once you enter the correct password?
 - Once the alarm has been turned off, can you set it off again by entering a certain number of incorrect passwords?
- **Optional:** Set a new password and enter the new password.
 - Does the green LED turn on?

Note: Include your video with your finished lab.

Alternatively, if you are unable to record video, record your observations below:

1.7 Conclusion

1-44 Describe the differences between a mechanical and digital safe. Explain when you would use a digital safe over a mechanical safe and vice versa (you cannot use the examples in the Introduction section).

1-45 What was one unexpected challenge you came across? How did you overcome it?

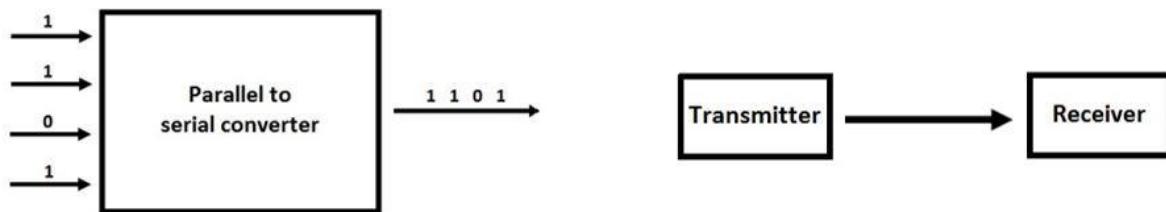
1-46 What was the purpose of the shift registers in the circuit?

1-47 How would the circuit change if you built it in Multisim instead of PLD?

1-48 If you could re-build the circuit, what would you change? Would you approach the problem differently?

Lab Manual: Digital Electronics

Using the Digilent Digital Electronics Board for NI ELVIS III



Left: *Parallel-in Serial-out Transmitter*

Right: *Serial Transmitter and Receiver*

Lab 18: Digital Communications – Application
Lab #3

© 2018 National Instruments

All rights reserved. Neither this resource, nor any portion of it, may be copied or reproduced in any form or by any means without written permission of the publisher.

National Instruments respects the intellectual property of others, and we ask our readers to do the same. This resource is protected by copyright and other intellectual property laws. Where the software referred to in this resource may be used to reproduce software or other materials belonging to others, you should use such software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

LabVIEW and National Instruments are trademarks of National Instruments.

All other trademarks or product names are the property of their respective owners.

Additional Disclaimers: The reader assumes all risk of use of this resource and of all information, theories, and programs contained or described in it. This resource may contain technical inaccuracies, typographical errors, other errors and omissions, and out-of-date information. Neither the author nor the publisher assumes any responsibility or liability for any errors or omissions of any kind, to update any information, or for any infringement of any patent or other intellectual property right.

Neither the author nor the publisher makes any warranties of any kind, including without limitation any warranty as to the sufficiency of the resource or of any information, theories, or programs contained or described in it, and any warranty that use of any information, theories, or programs contained or described in the resource will not infringe any patent or other intellectual property right. THIS RESOURCE IS PROVIDED "AS IS." ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, ARE DISCLAIMED.

No right or license is granted by publisher or author under any patent or other intellectual property right, expressly, or by implication or estoppel.

IN NO EVENT SHALL THE PUBLISHER OR THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, COVER, ECONOMIC, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS RESOURCE OR ANY INFORMATION, THEORIES, OR PROGRAMS CONTAINED OR DESCRIBED IN IT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF CAUSED OR CONTRIBUTED TO BY THE NEGLIGENCE OF THE PUBLISHER, THE AUTHOR, OR OTHERS. Applicable law may not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Lab 18: Digital Communications – Application Lab #3

In this lab, you will utilize a variety of concepts and skills developed in the digital electronics lab series to design and build a digital communication device. The skills and concepts you will use include flip-flops, counters, shift registers and semiconductor memory. The communication system will be able to display messages along with a variety of possible extensions you may choose to include as part of your design.

Learning Objectives

In this lab, students will:

1. Build a transmitter and/or receiver to create a complete communication system.
2. Explore the similarities and differences between parallel and serial communication.
3. Explore how flip-flops can be combined to makeshift registers.
4. Learn how to use a Function Generator to simulate clock pulse.
5. Have the opportunity to use a standard character-encoding scheme or create their own to send more complicated messages.
6. Have the opportunity to explore the process of message encryption and decryption.

Required Tools and Technology

Platform: NI ELVIS III

Instruments used in this lab:

- Function Generator

Note: The NI ELVIS III Cables and Accessories Kit (purchased separately) is required for using the instruments

- ✓ View User Manual:
<http://www.ni.com/en-us/support/model.ni-elvis-iii.html>
- ✓ View Tutorials:
https://www.youtube.com/playlist?list=PLvcPluVaUMIWm8ziaSxv0gwtshBA2dh_M
- ✓ Install Soft Front Panel support:
<http://www.ni.com/documentation/en/ni-elvis-iii/latest/getting-started/installing-the-soft-front-panel/>

Hardware: Digilent Digital Electronics Board for NI ELVIS III

- ✓ View NI Digital Electronics Board Manual:
<http://www.ni.com/pdf/manuals/376627b.pdf>

Hardware: Electrical Components

- Wire

Software: NI Multisim 14.0.1 Education Version or newer

- ✓ Install Multisim:
http://www.ni.com/gate/gb/GB_ACADEMICEVALMULTISIM_US
- ✓ View Help:
<http://www.ni.com/multisim/technical-resources/>

Software: NI LabVIEW FPGA Vivado 2014.4

- ✓ Install:
<http://www.ni.com/download/labview-fpga-module-2015-sp1/5920/en/>

Note: Digilent Driver (The installer above automatically downloads the installer below onto your computer)

- ✓ Navigate to:

C:\NIFPGA\programs\Vivado2
014_4\data\xicom\cable_driver
s\nt64\digilent
✓ Install: install_digilent.exe

Expected Deliverables

In this lab, you will collect the following deliverables:

- Screenshot of your finished circuit
- [Optional] Sketches and screenshots of your encryption and decryption circuits
- Video of your working circuit
- Screenshots of your circuit after any change
- Conclusion questions

Your instructor may expect you to complete a lab report. Refer to your instructor for specific requirements or templates.

1.1 Theory and Background

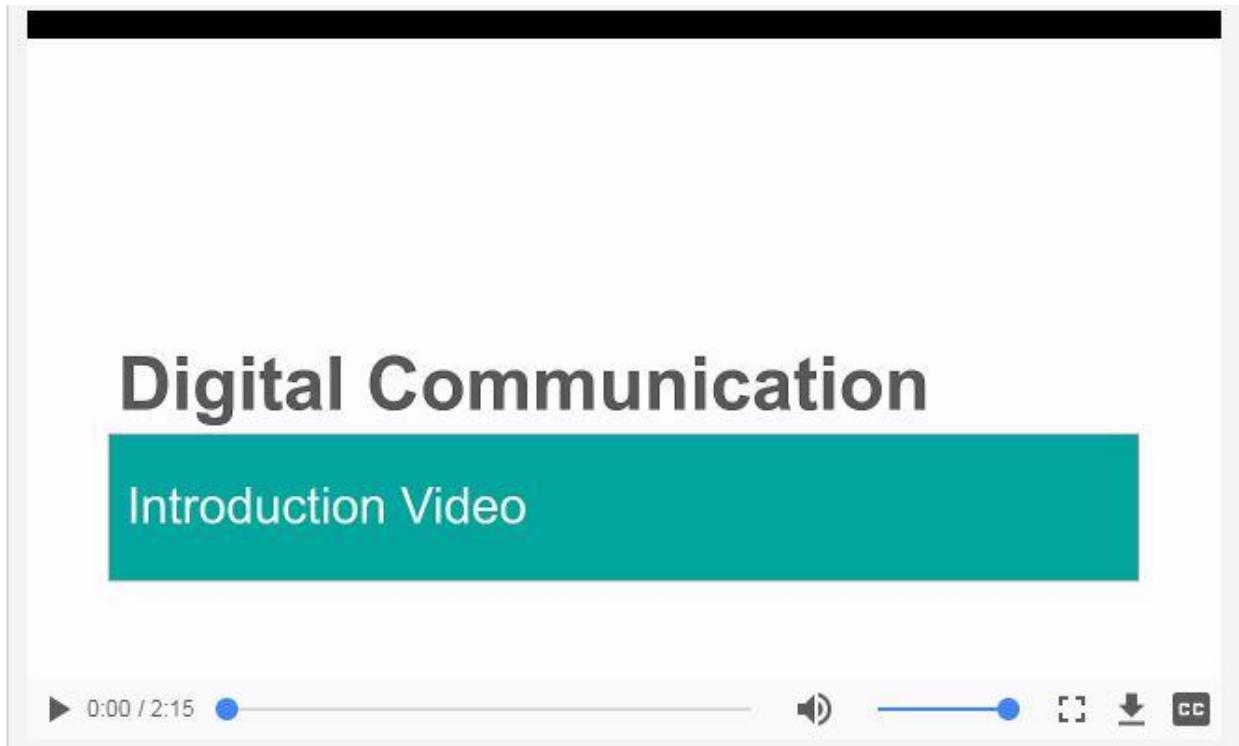


Figure 1-1 Video View the video here: <https://youtu.be/HKjR2VnWFqw>

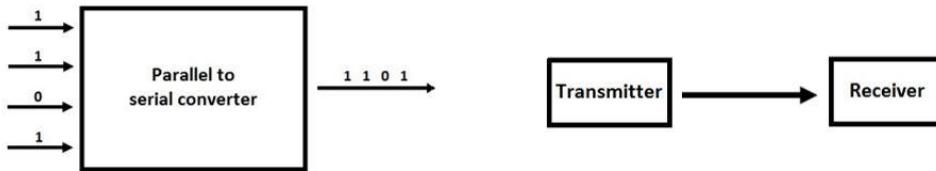


Video Summary

- This video will explore how many of the concepts and skills you have learned in this course will help you design and build a digital communication device
- Communication devices send and receive messages by methods of digital communication
- Communication can be done in series or parallel
- Encryption is a security measure that manipulates digital messages to prevent those without a decryption key from reading the messages

Series and Parallel Communication

Serial communication is a method of *digital communication* that sends individual bits through a single communication channel one at a time, thus creating a long stream of data. Often, serial transmitters will receive all bits of a message simultaneously, then convert the multiple inputs into a single stream of data for transmission.

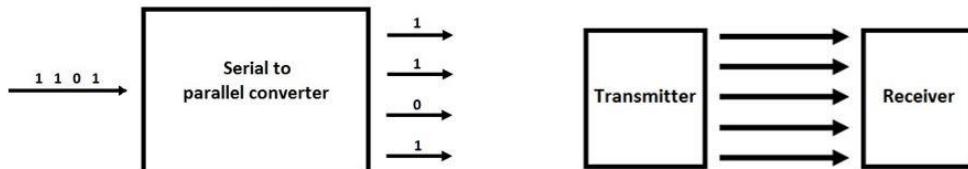


Left: *Parallel-in Serial-out Transmitter*

Right: *Serial Transmitter and Receiver*

Figure 1-2 PISO transmitter and serial transmitter and receiver

In contrast, *parallel communication* sends each bit of data from a message through its own channel simultaneously. In parallel systems, each bit of information requires its own line. For example, if an 8-bit message is being sent, the transmitter would need to be connected to the receiver with eight channels.



Left: *Serial-in Parallel-out Transmitter*

Right: *Parallel Transmitter and Receiver*

Figure 1-3 SIPO transmitter and parallel transmitter and receiver

Because parallel communication requires multiple lines to transmit data, it requires more material, which makes it the more expensive way to communicate over long distances. For this reason:

- Serial communication is more common for long-distance transmission.
- Parallel communication is often used when data needs to travel a very short distance, like in computer RAMs and integrated circuits.

Though both serial and parallel communications have their advantages, serial is becoming more and more popular in the majority of fields (such as electronics and computers). This is because as technology advances, serial communication's transmission speed is increasing and its ability to preserve signal integrity is greater than that of parallel communication.

Encryption and Decryption

Encryption is the process of manipulating a digital message with software algorithms to prevent those without a decryption key from reading it. Encryption is a method of security commonly used by the military and the government to decrease the odds of intercepted classified information being read.

To decrypt a message, you need to know the *decryption key* created by the person who encrypted the message. The correct decryption key will manipulate the encrypted message in such a way that will return it to the original message. If you choose to add encryption and decryption to this lab, you will use an 8-bit binary key and a series of logic gates to both encrypt and decrypt the message.

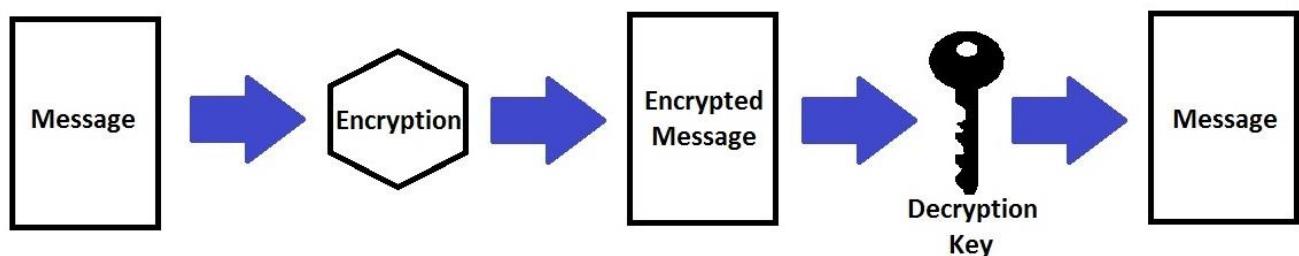


Figure 1-4 Encryption process

In this lab, your goal is to build a circuit that generates, sends, receives, and displays serial 8-bit messages. Ideally you will communicate between two Digital Electronics Boards, but you can both send and receive the message on one board if needed. In addition, you will be using the software Function Generator to generate a clock pulse. It will replace the system clock of the Digital Electronics Board which is set too high at a frequency of 128 MHz.

You may also add the ability to communicate simple words and phrases back and forth with the help of a *character-encoding scheme*. Additionally, you can add a level of security to your messages using encryption and decryption. You can also transmit the message between two different Digital Electronics Boards. Finally, instead of 8-bit messages, you can also modify your circuit to transmit numbers as 7-bit signals, corresponding to the segments of an SSD.

Before starting this lab, it is suggested that you review the following labs from previous weeks:

- Lab 10: Flip-Flops
- Lab 11: Counters
- Lab 13: Shift Registers
- Lab 14: Semiconductor Memory

Research and Planning

- Research and summarize the following keywords to help you gain a better understanding of what you will need in this lab.
- You will be able to come back and add more information at any point in the lab.

1-1 Data transmission:

1-2 Telecommunication:

1-3 Serial Communication:

1-4 Parallel communication:

1-5 Shift register:

1-6 Flip-flop (the different types available in Multisim and how they work):

1-7 Counter (types available in Multisim and the output type, clear and clock functions)

1-8 Function Generator:

Answer the following questions to help prepare you for the building process. These questions will also ensure your research covered the necessary information before you start building your communication system.

1-9 Make a checklist of all the different functions your circuit should be able to do. Include any additional features you plan to add.

1-10 Describe the differences between serial and parallel communication. List the advantages and disadvantages of each and explain why serial communication is being used in this lab.

1-11 How would you design your circuit to allow outputting multiple 8-bit binary messages?

1-12a The transmitter will be parallel-in/serial-out. Explain how a flip-flop works and determine how many flip-flops do you need to send 8-bit characters (8)?

1-12b What kind of flip-flops do you need?

1-13 In this lab, you use the Function Generator to control the frequency input of your clock. What are advantages of using the Function Generator compared to just modifying the clock built in the Digital Electronics Board?

1.2 Exercise: Building the Communications System

The following block diagram shows how all the different parts of your communication system will be connected. Use this as a guideline for connecting the different components detailed in the instructions below.

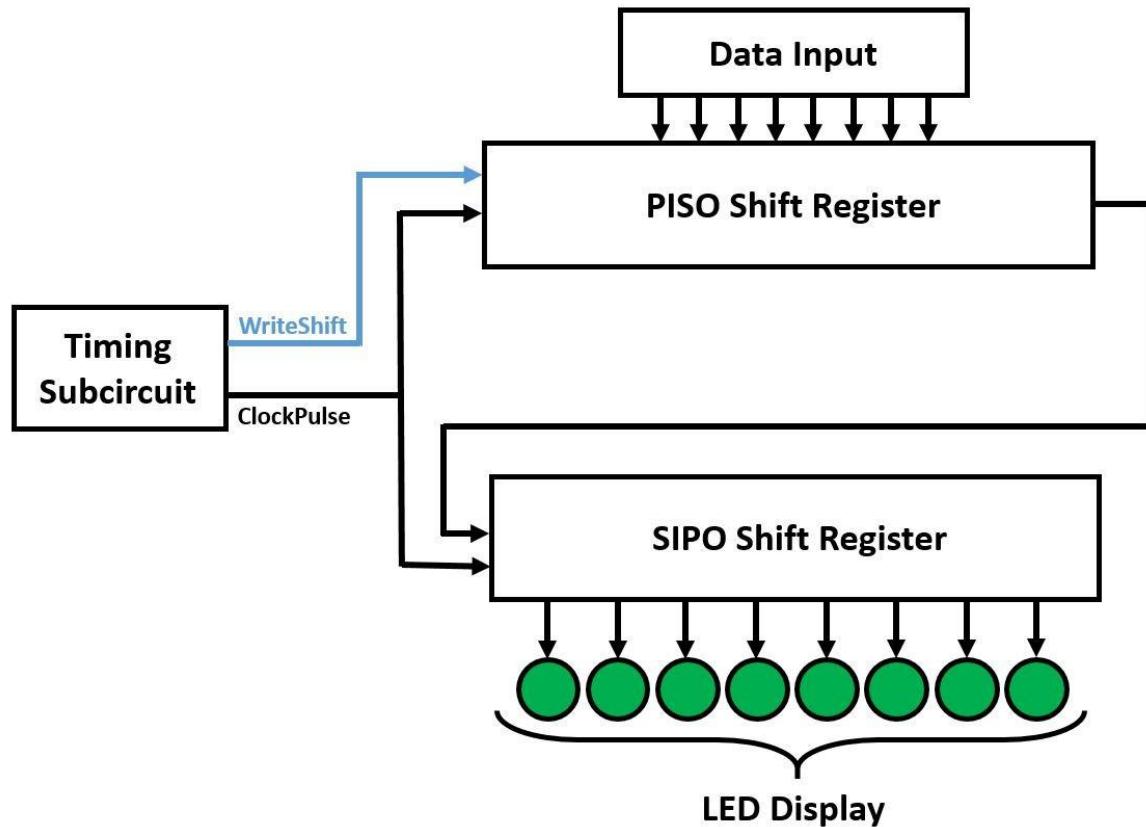


Figure 1-5 Communication system block diagram

Getting Started

1. Open a **NEW PLD** file.
2. Create the following PLD subcircuits in the main PLD workspace: **Timing**, **PISO Shift Register**, and **SIPO Shift Register**.

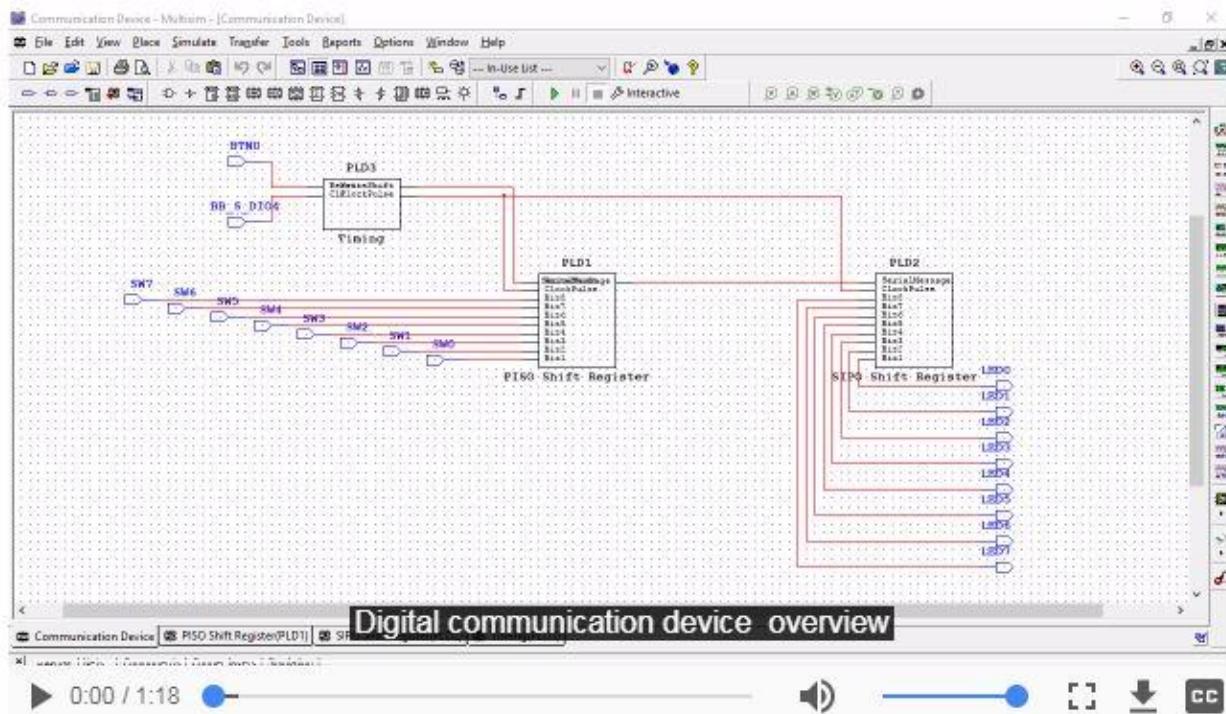


Figure 1-6 Communication system video View the video here: https://cf-ts.mythinkscape.com/video/Digital_Communication_Overview_no_cursor.mp4

Components Overview

- Timing Sub-Circuit:
<https://mythinkscape.com/labs/v2/23305/steps/21204#Timing%20SubC>
- PISO Shift Register (Transmitter) Sub-circuit:
<https://mythinkscape.com/labs/v2/23305/steps/21204#PISO%20SubC>
- SIPO Shift Register (Receiver) Sub-circuit:
<https://mythinkscape.com/labs/v2/23305/steps/21204#SIPO%20SubC>
- Input and Output Connectors:
<https://mythinkscape.com/labs/v2/23305/steps/21204#I/O%20Connectors>

Timing Sub-circuit

3. In the Timing sub-circuit, place the following components:
 - (1) binary counter (CNTR_7BIN)

Note: This ensures the right number of clock pulses are sent to the PISO shift register.

- (1) D flip-flop (FF_D_CO)

Note: This controls the WriteShift output.

- (3) INV gates, (1) 3-input AND gate, and (1) OR gate

4. Place and name the following connectors

- **Clk** and **Reset** inputs
- **ClockPulse** and **WriteShift** outputs

5. Wire the timing sub-circuit as shown in the image below:

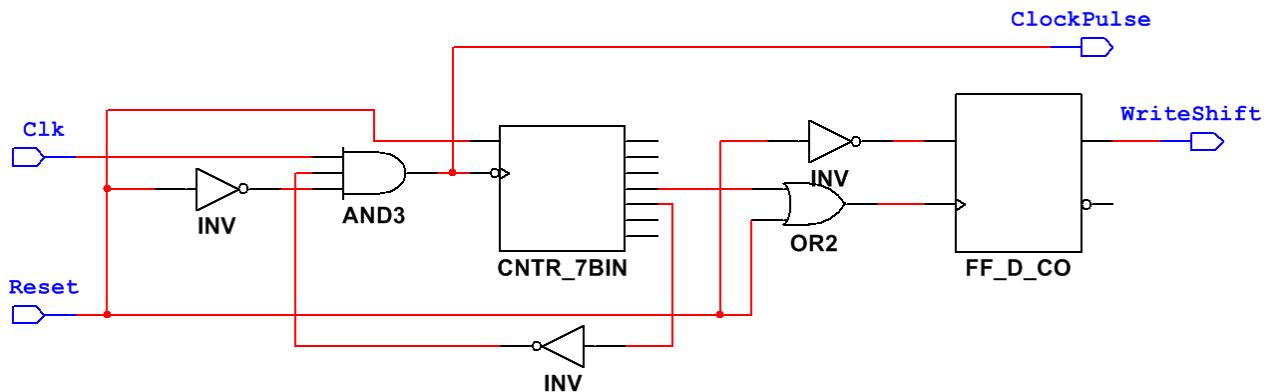


Figure 1-7 Timing sub-circuit

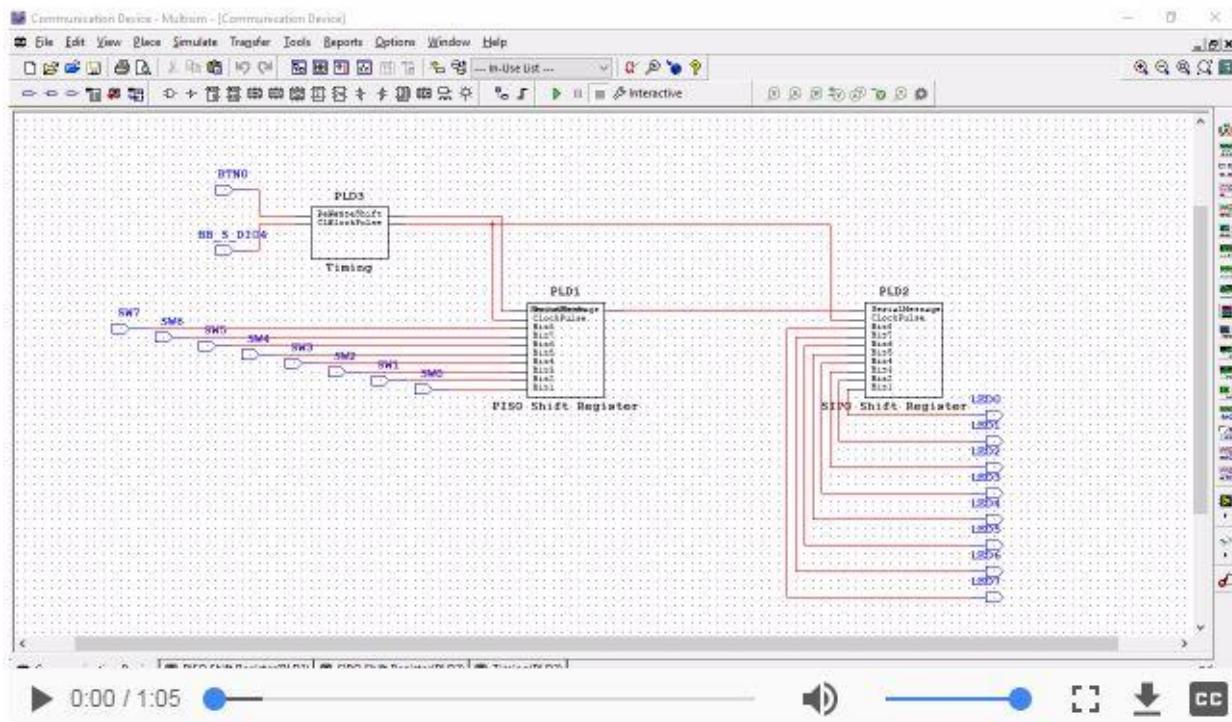


Figure 1-8 Timing sub-circuit video View the video here: https://cf-ts.mythinkscape.com/video/Timing_no_cursor.mp4

PISO Shift Register (Transmitter) Sub-circuit

6. In the **PISO Shift Register** sub-circuit, place the following components:
 - o (1) inverter (INV)
 - This ensures the proper functioning of the circuit.
 - o (21) 2-input NAND gates
 - These gates organize the message data and make sure it is correctly inserted in the flip-flops.
 - o (8) D flip-flops (FF_D_CO)
 - The flip-flops hold the message data so that it can be transmitted in correct order.
7. Place and name the following connectors:
 - o Flip-flop inputs: **Bit1 - Bit8**
 - o Shift register output: **SerialMessage**
 - o Clock pulse input: **ClockPulse**
 - o Write/shift input: **WriteShift**
8. Wire the 8-bit PISO shift register as shown in the image below:

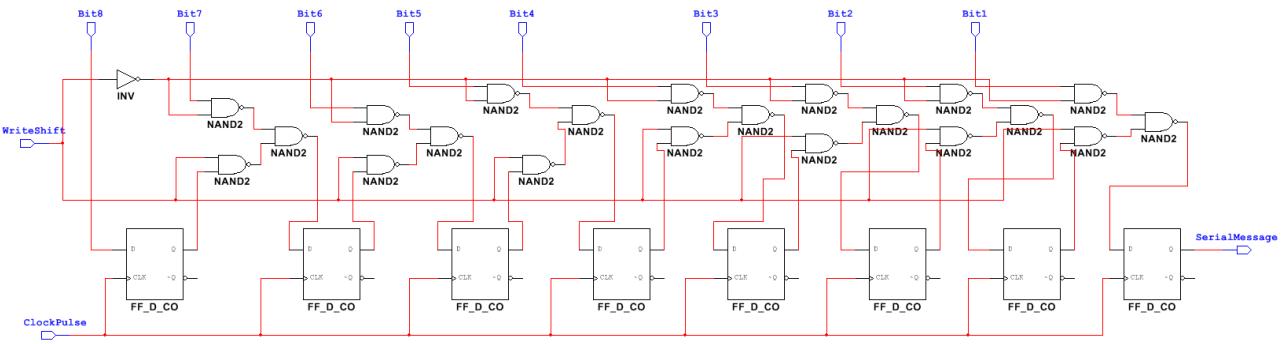


Figure 1-9 8-bit PISO shift register circuit diagram

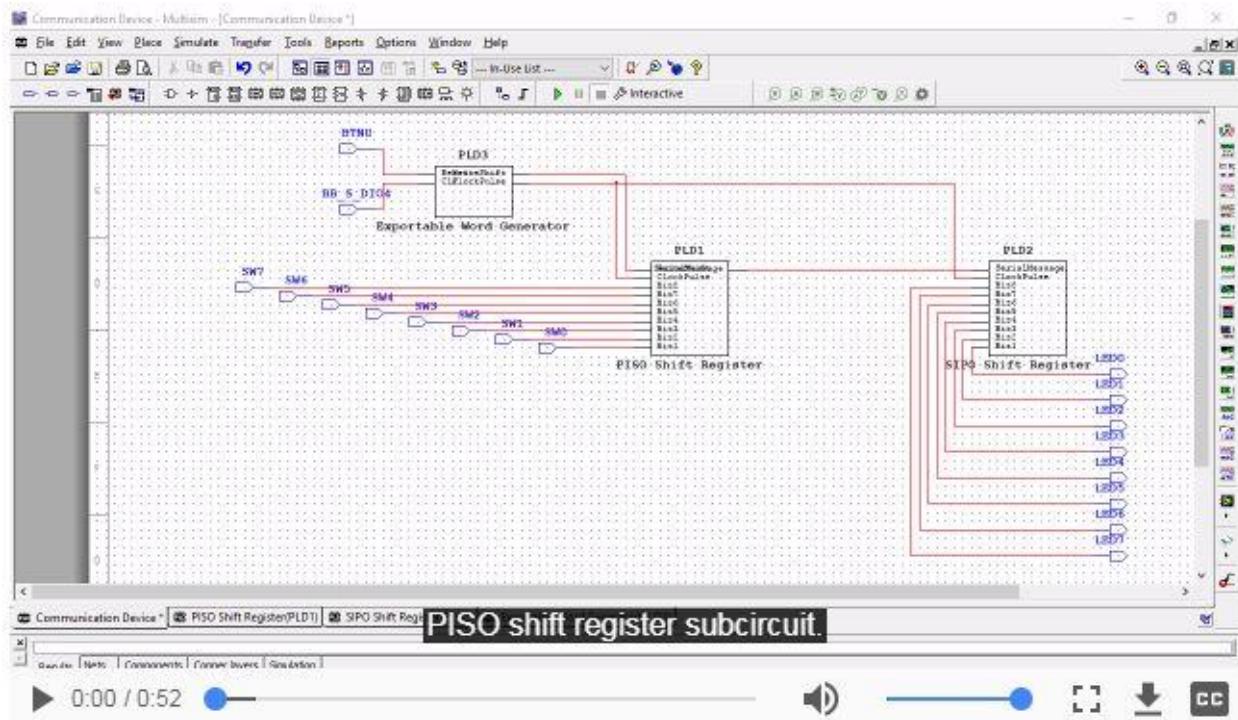


Figure 1-10 8-bit PISO shift register circuit video View the video here: https://cf-ts.mythinkscape.com/video/PISO_Shift_register_no_cursor.mp4

SIPO Shift Register (Receiver) Sub-circuit

9. In the **SIPO Shift Register** sub-circuit, place 8 D flip-flops (FF_D_CO). These flip-flops receive the transmitted data in the correct order and hold the values so the message can be read.
10. Place and name the following connectors:
 - o Flip-flop outputs: **Bit1 - Bit8**
 - o Shift register input: **SerialMessage**
 - o Clock pulse input: **ClockPulse**
11. Wire the 8-bit SIPO shift register as shown in the image below:

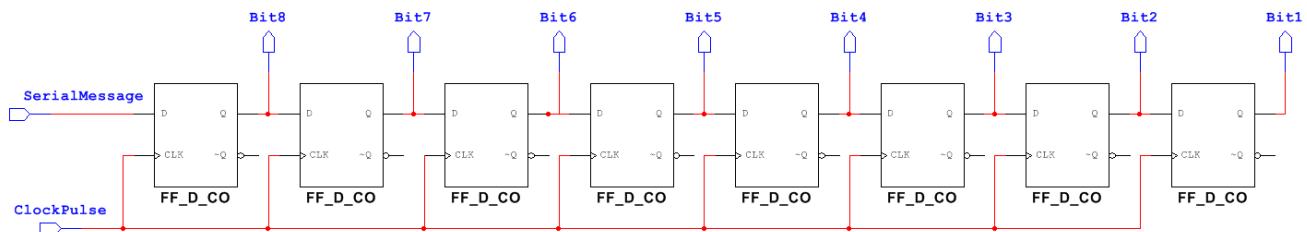


Figure 1-11 8-bit SIPO shift register circuit diagram

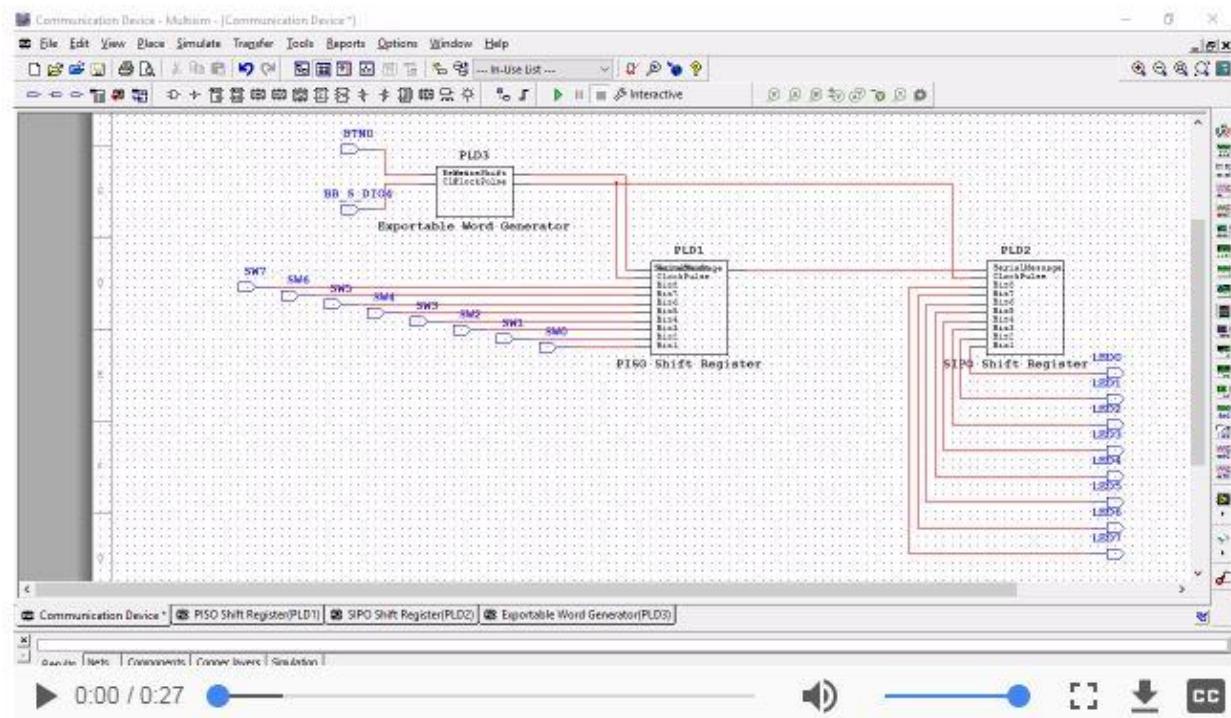


Figure 1-12 8-bit SIPO shift register circuit video View the video here: https://cf-ts.mythinkscape.com/video/SIPO_Shift_Register_no_cursor.mp4

Input and Output Connectors

12. In the main circuit, add an **LED connector** to each output of the SIPO shift register. LED0 should display Bit1, LED1 should display Bit2, etc.
13. Add 8 switch connectors (**SW0 - SW7**) to each input of the PISO shift register. SW0 should control Bit 1, SW1 should control Bit2, etc.
14. Add the **BTN0** and **BB_S_DIO4** inputs and connect them to the Timing sub-circuit as shown in the image below.

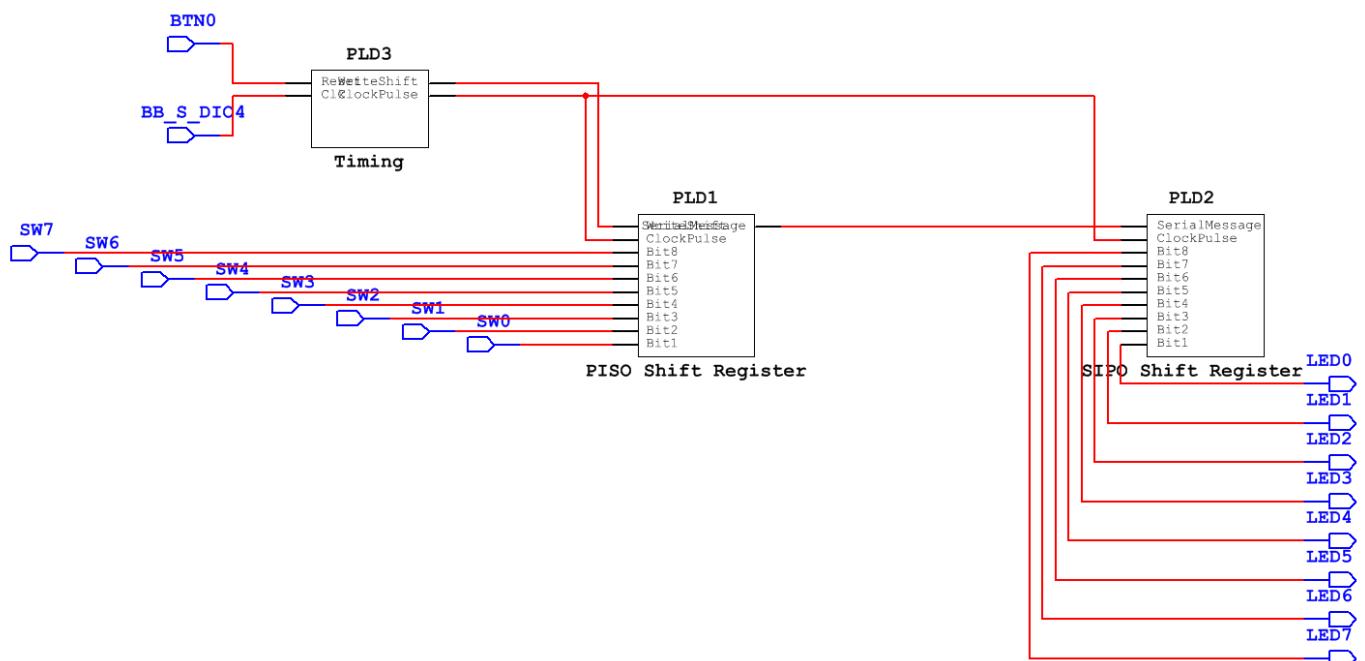


Figure 1-13 8-bit SIPO shift register circuit diagram

1.3 Exercise: Setting Up the Function Generator

Instructions:

1. Connect the BNC plug to the **FGEN port** of the NI ELVIS III platform. Connect the red clip to a wire and plug the wire into the **BB_S_DI04**. Similarly, connect the black clip to a wire and plug it into the **GND**.
2. Set the following values to the Function Generator:
 - o Waveform: **Square**
 - o Frequency: **1,024 Hz**
 - o Amplitude: **3.00 Vpp**
3. Make sure that the Device (**NI ELVIS III**) and Signal Route (**FGEN BNC**) are successfully recognized by the Function Generator. In order to be recognized, the Elvis Board must be turned on.

Use the following image as a reference for the settings:

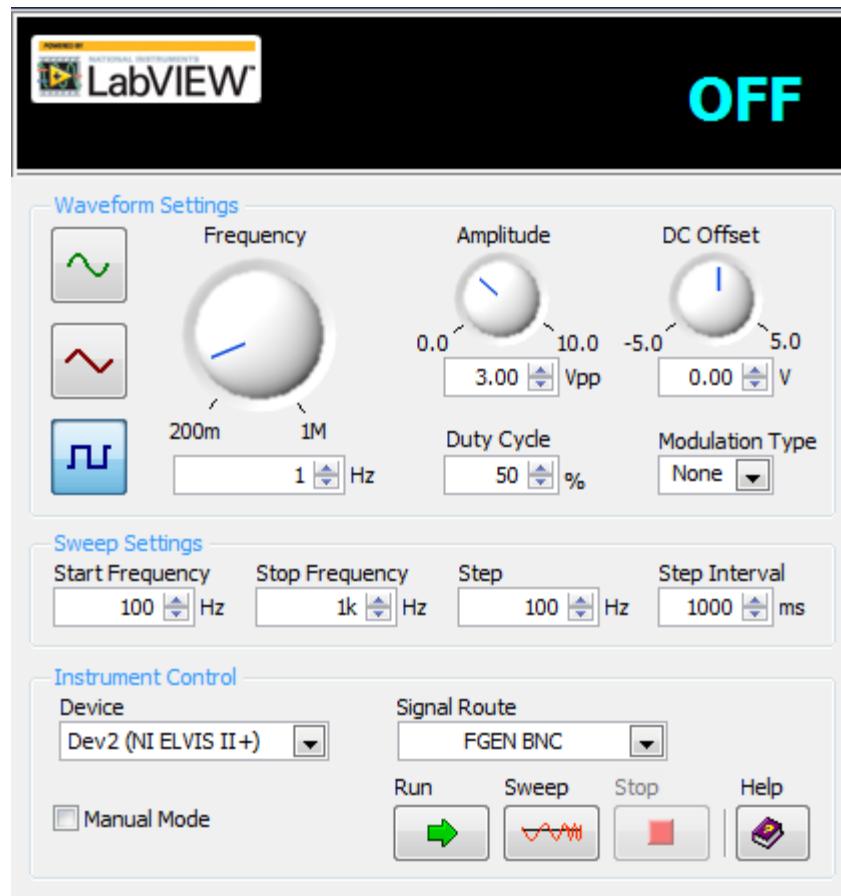


Figure 1- 14 Settings Reference

1.4 Implement: Putting It All Together

1. Using the block diagram given at the beginning of this section, *Figure 1-5*, wire the different sub-circuits together.
2. Export the circuit to the Digital Electronics Board. Make sure you restart the board before you export your circuit so that the changes take effect.

Note: Take a screenshot, draw a sketch, or take a photo of your finished circuit and all of the sub-circuits you built and include it with your finished lab.

1.5 Testing and Troubleshooting

This section walks you through a series of steps to test that your communication system is working properly. Use the troubleshooting guide at the end of this section to help fix any issues you come across during the test.

Tip: add interactive digital constants and probes to your circuit in PLD so that you can test fixes you've made to your circuit without having to export it to the Digital Electronics Board.

Instructions:

1. Make sure your PLD design has been properly exported to the Digital Electronics Board.
2. What is the message you will be sending? Configure the switches on the board accordingly.
3. Start the Function Generator.
4. Record the LED display outputs. The display should pause once your message is displayed. Does the sent message match the received message?
5. Re-configure the switches to set up a different message, then press the reset button. Does the LED display output your new message?

Things to always check:

- Is the Digital Electronics Board turned on?
- Did you turn your Digital Electronics Board off and on before deploying it?
 - The board needs to be turned off and on to reset

The PLD circuit won't deploy to the Digital Electronics Board:

- Ensure you have turned on the NI ELVIS III and the Digital Electronics Board.
- Ensure that the board is connected to your computer and to power. Sometimes the USB port on your computer isn't making a good connection, so try using a different one.
- Be sure to select the correct board when prompted.
- Be sure to select the correct Xilinx tool (either 32-bit or 64-bit depending on your operating system).
- Check that the correct connectors have been chosen in PLD. Their names must match up with those on the Digital Electronics Board. Also, make sure that they are correctly wired in PLD and on the board.

No message is received, or only one output (either 0 or 1) is received:

- Make sure the **WriteShift** output of the Timing circuit is connected to the **WriteShift** input of the transmitter.

- Make sure that you've configured the switches on the Digital Electronics Board properly.
- Make sure that you've wired the different components together correctly.

The received message does not match the transmitted message:

- Make sure you're reading the transmitted and received messages in the same direction.
- Check that the counter is stopping both the transmitter **AND** the receiver at the proper time (after 16 clock pulses).
- Check that **WriteShift** is being activated after the first 8 clock pulses.
- Check that both shift registers are built properly.

The circuit no longer displays messages correctly after being reset:

- Check the wiring of the **Reset** pin inside the Timing sub-circuit.
- Make sure that activating Rest changes the **WriteShift** output back to a low.

Note: if you made any changes to your circuit in this section, take screenshots, draw sketches, or take photos of your new circuit and include them with your finished lab.

1.6 Extensions

This lab contains four extension activities. Choose *ONE or more* of these extensions to add to your digital communication system:

- Message Encryption and Decryption:
<https://mythinkscape.com/labs/v2/23305/steps/21217#Message%20Encryption%20Detection>
- Character Encoding Scheme:
<https://mythinkscape.com/labs/v2/23305/steps/21217#Character%20Encoding%20Scheme>
- Communicating Between Two Digital Electronics Boards:
<https://mythinkscape.com/labs/v2/23305/steps/21217#Communicating%20Between%20Two%20DSDB%20Boards>
- Seven-Segment Display:
<https://mythinkscape.com/labs/v2/23305/steps/21217#SSD>

Message Encryption and Decryption

This section discusses an addition you can make to the communication system you've already built. The message encryption and decryption can protect your message from being read by someone without the decryption key. This section covers the research, planning, and construction of this additional feature.

Research and summarize the following keywords to help you gain a better understanding of what you will need in this part of the lab. You will be able to come back and add more information at any point in the lab.

1-14 Encryption:

1-15 Decryption:

1-16 Decryption Keys:

Answer the following questions to help prepare you for the building process. These questions will also ensure your research covered the necessary information before you start building your communication system.

1-17 What should the final circuit be able to accomplish?

1-18 To add encryption and decryption to your communication, each bit input to the transmitter and output from the receiver needs to pass through a logic gate along with a key. To choose a logic gate, consider their different truth tables. You want a logic gate that gives: $A + B = C$ to encrypt, then $C + B = A$ to decrypt. Where **A** is your plain message, **B** is your key, and **C** is your encrypted message (as shown in the image below). With this in mind, which logic gate would you choose?

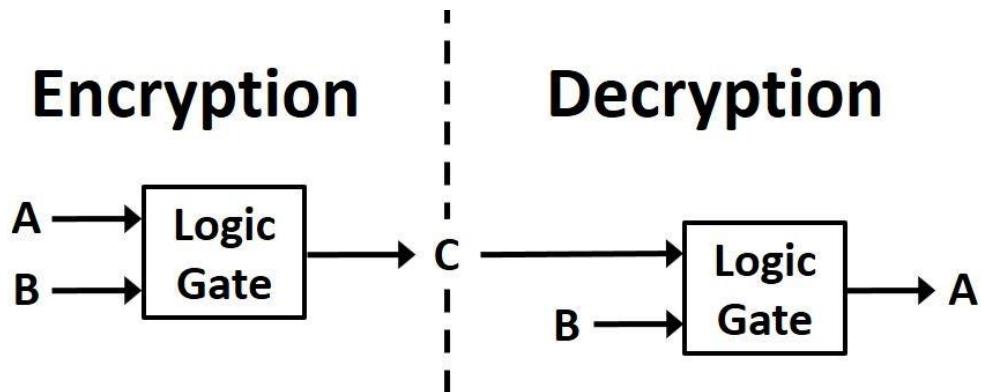


Figure 1-15 Encryption and decryption logic

1-19 Between which components would you add the encryption?

1-20 Between which components would you add the decryption?

- Sketch your *encryption circuit* and show how it will connect to what you have already built.
- Sketch your *decryption circuit* and show how it will connect to what you have already built.

Note: Include your sketches with your finished lab.

Use the following instructions to build the addition(s) to your original circuit:

1. In the PISO shift register, place a logic gate between the message bit inputs and the inputs to the NAND gates.
2. Place a logic gate between the output of the D flip-flops in the SIPO shift register and the output pins. One input to the gate will be a bit from the key, the other input is a bit from the flip-flops. The output of the gate goes to the output pins.

Note: Make sure that the bits of the key are aligned the same way with both the transmitter and receiver.

Note: Make sure that you enter the key using Digital Highs and Digital Lows instead of Interactive Digital Constants.

Note: Once you've built the new additions into the circuit, take screenshots of each and include them with your finished lab.

Testing and Troubleshooting

1. Connect digital probes to the output of the SIPO shift register to substitute LED display on Multisim.
2. Enter the correct decryption key and observe if the correct message is displayed.
3. Enter an incorrect decryption key and observe if the message is displayed.

Use the following question to help you troubleshoot and fix any issues with your circuit.

The received message does not match the transmitted message.

- Make sure that the encryption key and the decryption key match and are in the same order.

The message is not displayed when the correct decryption is entered.

- Make sure that the logic gates between the message bit lines and the PISO shift register and between the SIPO shift register and the LED display are connected properly.

Note: If you made any changes to your circuit in this section, take screenshots of your new circuit and upload them below.

If you would like to add another extension, follow this link:
<https://mythinkscape.com/labs/v2/23305/steps/21217> .

If you are done adding all the extensions, proceed to *1.7 Exercise: Final Testing*.

Character-Encoding Scheme

The following section discusses an addition you can make to the communication system you've already built. The character-encoding scheme allows you to send more meaningful messages. This section covers the research, planning, and construction of this additional feature.

Research and summarize the following keywords to help you gain a better understanding of what you will need in this part of the lab. You will be able to come back and add more information at any point in the lab.

1-21 Character Encoding Schemes:

1-22 ASCII:

Answer the following questions to help prepare you for the building process. These questions will also ensure your research covered the necessary information before you to start building your communication system.

1-23 What are the pros and cons of using a character-encoding scheme for your message?

1-24 You can choose to use either the standard character-encoding scheme ASCII or you can create your own. Which character-encoding scheme are you going to use?

1-25 If you are using a predetermined coding scheme find the conversion table online and have it available for the continuation of the lab. If you are creating your own scheme build your own conversion table for the alphabet.

Note: Include your scheme with your finished lab.

1-26 Will you have to make any changes to the pre-existing circuit to implement your encoding scheme?

1-27 How do you represent the letters A, J, T, and Z using your chosen character-encoding scheme?

1-28 How would you represent the message “hello” using your chosen character-encoding scheme?

Use the following instructions to implement the addition(s) in your original circuit.

You can reference the following chart for character conversion:

ASCII Code: Character to Binary

0	0011 0000	o	0100 1111	m	0110 1101
1	0011 0001	p	0101 0000	n	0110 1110
2	0011 0010	q	0101 0001	o	0110 1111
3	0011 0011	r	0101 0010	p	0111 0000
4	0011 0100	s	0101 0011	q	0111 0001
5	0011 0101	t	0101 0100	r	0111 0010
6	0011 0110	u	0101 0101	s	0111 0011
7	0011 0111	v	0101 0110	t	0111 0100
8	0011 1000	w	0101 0111	u	0111 0101
9	0011 1001	x	0101 1000	v	0111 0110
A	0100 0001	y	0101 1001	w	0111 0111
B	0100 0010	z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	,	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	i	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(0010 1000
N	0100 1110	l	0110 1100)	0010 1001
				space	0010 0000

Figure 1-16 ASCII code

Testing and Troubleshooting

- Convert the binary output of the receiver into the corresponding characters. Does the sent message match the received message?

Use the following questions to help you troubleshoot and fix any issues with your circuit.

The received message does not match the transmitted message.

- Make sure that you're converting between binary numbers and characters correctly.
- Check that you have not made any inadvertent changes to your original circuit.

If you would like to add another extension, click [here](#).

If you are done adding all the extensions, proceed to *1.7 Exercise: Final Testing*.

Communicating Between Two Digital Electronics Boards

The following section discusses an addition you can make to the communication system you've already built. You will be transmitting messages between two different Digital Electronics Boards. This section covers the planning and construction of this additional feature.

Answer the following questions to help prepare you for the building process. These questions will also ensure your research covered the necessary information before you start building your communication system.

1-29 How would the circuit on the sending board differ from that on the receiving board?

1-30 What changes will you need to make to your original circuit to accommodate this extension?

1-31 How would the signal travel between the two boards? Where would it enter and exit them?

1-32 How will you coordinate the timing of the two boards?

Use the following instructions to build the addition(s) to your original circuit.

1. Modify the circuit for use on the sending board.
2. Modify the circuit for use on the receiving board.

Testing and Troubleshooting

- Send a message between two Digital Electronics Boards. Does the sent message match the received message?

Use the following questions to help you troubleshoot and fix any issues with your circuit.

The received message does not match the transmitted message.

- Check the wiring in your circuit.
- Ensure that the wire used to transmit the signal is connected to the proper input and output pins.
- Check that the two boards are receiving a clock signal that is synchronized (i.e. arriving from the same source).

Note: if you made any changes to your circuit in this section, take screenshots, draw a sketch, or take a photo of the new circuit and include them with your finished lab.

If you would like to add another extension, follow this link::

<https://mythinkscape.com/labs/v2/23305/steps/21217> .

If you are done adding all the extensions, proceed to *1.7 Exercise: Final Testing*.

Seven-Segment Display

The following section discusses an addition you can make to the communication system you've already built. You will be converting the 8-bit messages to numbers using the seven-segment display. This section covers the research, planning, and construction of this additional feature.

Research and summarize the following keywords to help you gain a better understanding of what you will need in this part of the lab. You will be able to come back and add more information at any point in the lab.

1-33 Seven-Segment Display:

Answer the following questions to help prepare you for the building process. These questions will also ensure your research covered the necessary information before you start building your communication system.

1-34 How many number digits do you want to display?

1-35 What are the differences between displaying the message in 8-bit and seven-segment display? What changes to your circuit do you have to make? Is the number of outputs the same?

1-36 How do you re-order the outputs for the seven-segment display to show numbers?

1-37 How would you output the signals from the FPGA chip to control the displays?

Use the following instructions to build the addition(s) to your original circuit:

1. Modify the circuit in order to display your messages in a seven-segment display.

Testing and Troubleshooting

- Convert the binary output of the receiver into their corresponding numbers. Does the sent message match the received message?

Use the following questions to help you troubleshoot and fix any issues with your circuit.

The received message does not match the transmitted message.

- Make sure that the switches and the SSD pins are wired to the correct shift register inputs and outputs.

Note: If you made any changes to your circuit in this section, take screenshots of your new circuit and upload them to the section below.

If you would like to add another extension, follow this link:
<https://mythinkscape.com/labs/v2/23305/steps/21217> .

If you are done adding all the extensions, proceed to *1.7 Exercise: Final Testing*.

1.7 Exercise: Final Testing

In this section, you will put your circuit through a final test to demonstrate that it works and answer some questions related to what you've built. Before you record your final circuit demonstration, make sure that you've gone through *Step 6 - Testing and Troubleshooting*. Your circuit should be fully functioning.

- Take a video of your circuit and go through the following tests:
 - Send a message that's 10 characters or longer from the transmitter to the receiver.
 - Did the receiver display the correct message?
 - [Optional]: If you used a character-encoding scheme, convert the binary output of the receiver back to characters. Does it match the sent message?
 - [Optional]: If you used a seven-segment display, convert the binary output of the receiver back to numbers. Does it match the sent message?

Note: Include your video with your finished lab.

1.8 Conclusion

1-38 Explain encryption and decryption and provide a scenario where it would be useful apart from those listed in the *Introduction* section.

1-39 What setting controls how fast the bits are sent from the transmitter to the receiver?

1-40 What would be different if you used parallel communication between the transmitter and receiver instead of serial communication? Compare the pros and cons of each.

1-41 How many possible decryption keys are there (remember that your key was 8-bits long)?

1-42 [Optional]: If you used a character-encoding scheme, why did you choose the scheme you used? Would you go back and change your decision?

1-43 What is one unexpected challenge you came across when doing this lab? How did you overcome it?

1-44 If you could re-build the circuit, what changes would you make? Would you approach them differently?
