CLC Tips and Tricks

Configurable Logic Cell (CLC) Tips and Tricks

Introduction

The Configurable Logic Cell (CLC) module provides programmable logic that operates outside the speed limitations of software execution. The CLC module accepts up to 256 input signals and through the use of configurable logic gates, reduces those inputs to four logic lines that drive one of the eight selectable single-output functions. The CLC peripheral allows the user to apply combinatorial and sequential logic to both internal and external signals with no CPU intervention through hardware-based logic, and allows users to automate task handling within the system.

Table of Contents

1. 4-to-2 Binary Encoder

Binary encoders allow users to take multiple digital inputs (2^n input bits) at one time and convert them into a single-encoded binary output (n output bits) using combinational logic. This can be useful in applications where there are several inputs that need to be monitored at the same time for changes, as the binary encoder circuit creates a shortened binary string that is representative of the logic state on each individual input signal. PIC® microcontroller devices equipped with the Configurable Logic Cell (CLC) module can be used to implement a binary encoder circuit using only CLCs, without the need for any additional hardware. The following example will demonstrate how two CLCs can be used to implement a 4-to-2 binary encoder in hardware using the PIC18F47Q43.

Table 1-1 contains the truth table as well as the corresponding Boolean expressions that were used to create the circuit needed for the 4-to-2 binary encoder. The resulting Boolean expressions show that the binary encoder can be implemented using two OR gates, in which the OR gate outputs provide the 2-bit binary encoded output string (the resulting circuit is illustrated by Figure 1-1). The four input signals, which in this example were standard GPIO pins, were configured using Peripheral Pin Select (PPS) as the CLC inputs to each OR gate based on the Boolean expressions below. The two CLC output signals, which are representative of the 2-bit encoded binary output, were also configured using PPS and are represented by the LEDs tied to pins RA5 and RA4. The MPLAB® Code Configurator (MCC) tool was used to setup the CLCs for this example, and the configurations for CLC1 and CLC2 are illustrated in Figure 1-2 and Figure 1-3, respectively.

Table 1-1. 4-to-2 Binary Encoder Truth Table

Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Equation 1-1. 4-to-2 Binary Encoder Boolean Expressions

$$A1 = Y3 + Y2$$

$$A0 = Y3 + Y1$$

Figure 1-1. 4-to-2 Binary Encoder Circuit

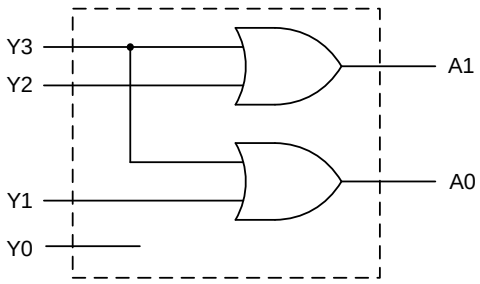


Figure 1-2. CLC1 Configuration for 4-to-2 Binary Encoder

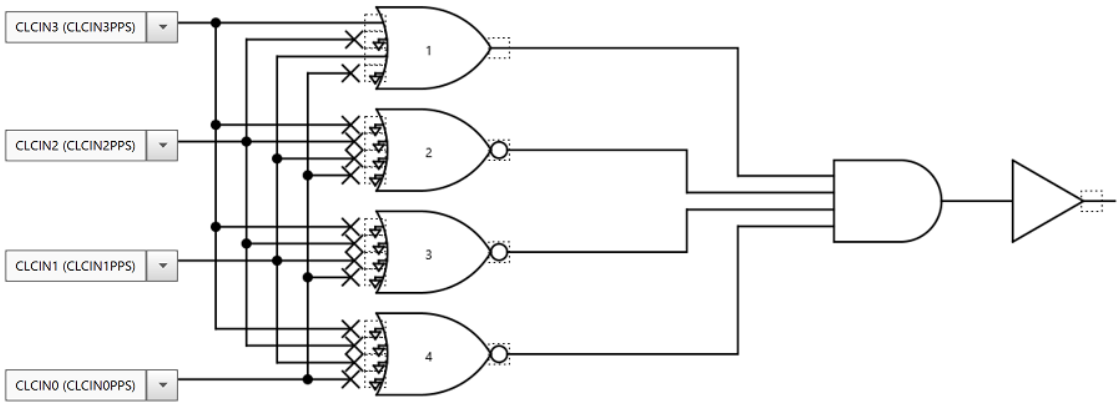
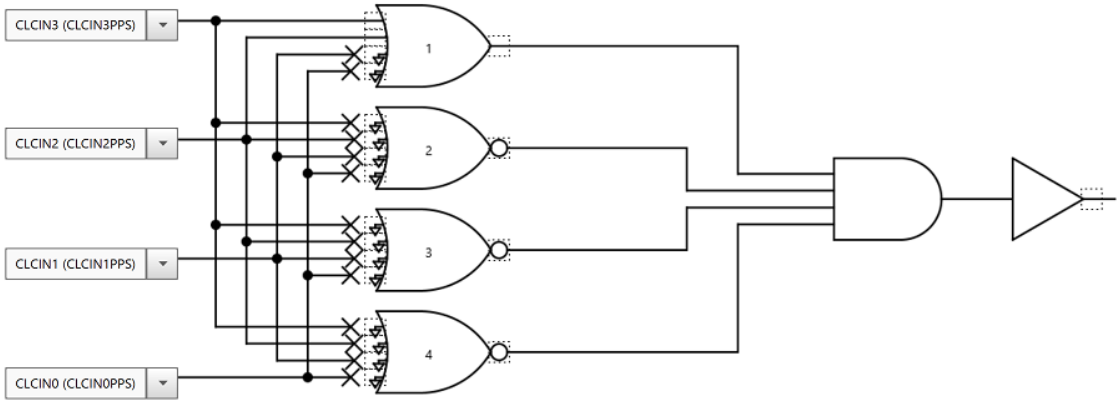


Figure 1-3. CLC2 Configuration for 4-to-2 Binary Encoder



Example 1-1. 4-to-2 Binary Encoder Initialization Code

```
/*This code block configures the CLCs
for 4-to-2 Binary Encoder.
  */

void CLC1_Initialize(void) {
    CLCSELECT = 0x00;         // SLCT 0
    CLCnPOL = 0x0E;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x03;          // D1S CLCIN3 (CLCIN3PPS)
    CLCnSEL1 = 0x02;          // D2S CLCIN2 (CLCIN2PPS)
    CLCnSEL2 = 0x01;          // D3S CLCIN1 (CLCIN1PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x22;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x00;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC1OUT 0
    CLCnCON = 0x82;           // EN enabled; INTN disabled; INTP disabled; MODE
4-input AND
}

void CLC2_Initialize(void) {
    CLCSELECT = 0x01;         // SLCT 1
    CLCnPOL = 0x0E;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x03;          // D1S CLCIN3 (CLCIN3PPS)
    CLCnSEL1 = 0x02;          // D2S CLCIN2 (CLCIN2PPS)
    CLCnSEL2 = 0x01;          // D3S CLCIN1 (CLCIN1PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x0A;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x00;          // CLCn Gate 1 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 1 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 1 Logic Selection
    CLCDATA = 0x00;           // CLC2OUT 0
    CLCnCON = 0x82;           // EN enabled; INTN disabled; INTP disabled; MODE
4-input AND
}
```

2. 8-to-3 Binary Encoder

In the previous example, two CLCs were used to implement a 4-to-2 binary encoder in hardware. The following example will demonstrate how the same principals can be applied to implement an 8-to-3 binary encoder using the PIC18F47Q43 and three CLCs. Table 2-1 below shows the truth table for the 8-to-3 binary encoder, and Figure 2-1 illustrates the resulting circuit that should be implemented using CLCs based on the derived Boolean expressions. The Boolean expressions derived from the truth table show that the 8-to-3 binary encoder can be implemented using three OR gates, each of which will be implemented using a CLC. The eight input signals in this example were configured using PPS as the CLC inputs to each OR gate based on the Boolean expressions below. The three CLC output signals, which represent the 3-bit encoded binary output, were also configured using PPS and are tied to pins RA6, RA5, and RA4. MCC was used to setup the CLCs for the 8-to-3 binary encoder, and the configurations for each CLC are illustrated in Figure 2-2, Figure 2-3 and Figure 2-4, respectively.

Table 2-1. 8-to-3 Binary Encoder Truth Table

Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	A2	A1	A0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Equation 2-1. 8-to-3 Binary Encoder Boolean Expressions

$$A2 = Y7 + Y6 + Y5 + Y4$$

$$A1 = Y7 + Y6 + Y3 + Y2$$

$$A0 = Y7 + Y5 + Y3 + Y1$$

Figure 2-1. 8-to-3 Binary Encoder Circuit

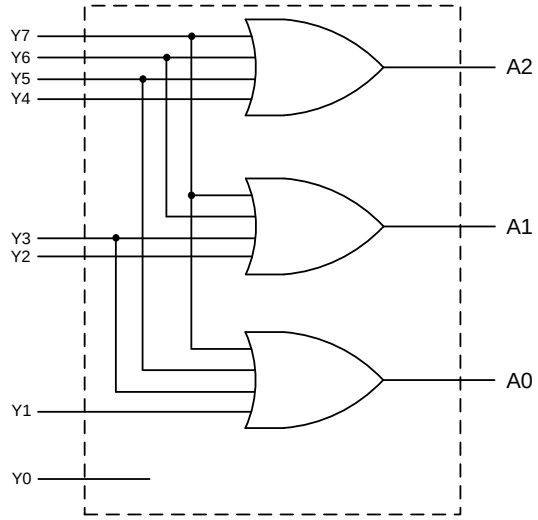


Figure 2-2. CLC1 Configuration for 8-to-3 Binary Encoder

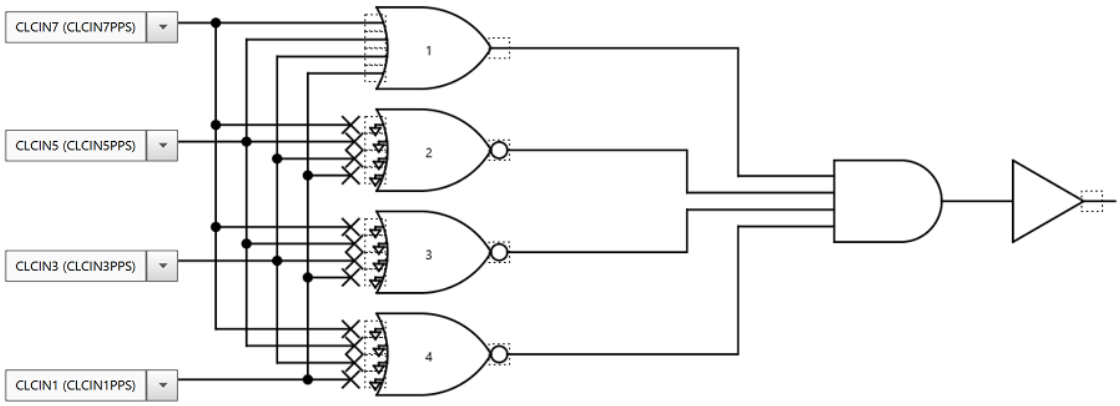


Figure 2-3. CLC2 Configuration for 8-to-3 Binary Encoder

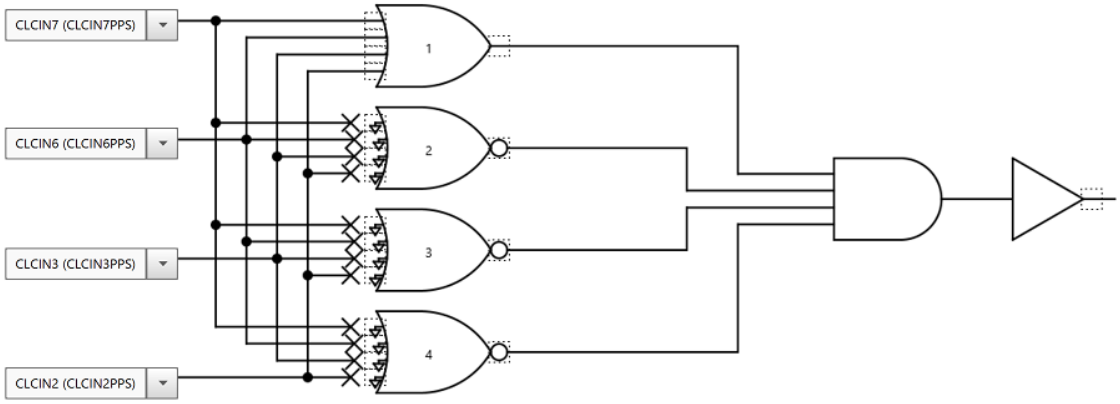
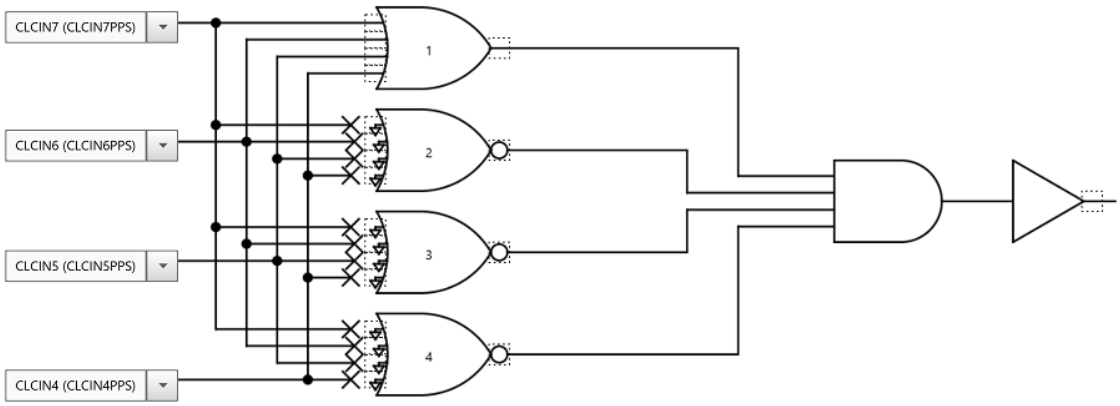


Figure 2-4. CLC5 Configuration for 8-to-3 Binary Encoder



Example 2-1. 8-to-3 Binary Encoder Initialization Code

```
/*This code block configures the CLCs
for 8-to-3 Binary Encoder.
  */

void CLC1_Initialize(void) {
    CLCSELECT = 0x00;         // SLCT 0
    CLCnPOL = 0x0E;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x07;          // D1S CLCIN7 (CLCIN7PPS)
    CLCnSEL1 = 0x05;          // D2S CLCIN5 (CLCIN5PPS)
    CLCnSEL2 = 0x03;          // D3S CLCIN3 (CLCIN3PPS)
    CLCnSEL3 = 0x01;          // D4S CLCIN1 (CLCIN1PPS)
    CLCnGLS0 = 0xAA;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x00;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC1OUT 0
    CLCnCON = 0x82;           // EN enabled; INTN disabled; INTP disabled; MODE
4-input AND
}

void CLC2_Initialize(void) {
    CLCSELECT = 0x01;         // SLCT 1;
    CLCnPOL = 0x0E;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x07;          // D1S CLCIN7 (CLCIN7PPS);
    CLCnSEL1 = 0x06;          // D2S CLCIN6 (CLCIN6PPS);
    CLCnSEL2 = 0x03;          // D3S CLCIN3 (CLCIN3PPS);
    CLCnSEL3 = 0x02;          // D4S CLCIN2 (CLCIN2PPS);
    CLCnGLS0 = 0xAA;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x00;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC2OUT 0
    CLCnCON = 0x82;           // EN enabled; INTN disabled; INTP disabled; MODE
4-input AND
}

void CLC5_Initialize(void) {
    CLCSELECT = 0x04;         // SLCT 4
    CLCnPOL = 0x0E;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x07;          // D1S CLCIN7 (CLCIN7PPS)
    CLCnSEL1 = 0x06;          // D2S CLCIN6 (CLCIN6PPS)
    CLCnSEL2 = 0x05;          // D3S CLCIN5 (CLCIN5PPS)
    CLCnSEL3 = 0x04;          // D4S CLCIN4 (CLCIN4PPS)
    CLCnGLS0 = 0xAA;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x00;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC5OUT 0
    CLCnCON = 0x82;           // EN enabled; INTN disabled; INTP disabled; MODE
4-input AND
}
```

3. 2-to-4 Binary Decoder

A binary decoder is a logic circuit that converts binary data from n inputs to 2^n outputs. This example will demonstrate how to implement 2-to-4 binary decoder using CLCs. Table 3-1 below shows the truth table for the 3-to-8 binary decoder, and Figure 3-1 illustrates the resulting circuit that should be implemented using CLCs, based on the derived Boolean expressions. MCC was used to setup the CLC modules for this application, and the configuration settings can be found in Figure 3-2, Figure 3-3, Figure 3-4 and Figure 3-5. The configuration code can be found in Example 3-1.

Table 3-1. 2-to-4 Binary Decoder Truth Table

A1	A0	Y3	Y2	Y1	Y0
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Equation 3-1. 2-to-4 Binary Decoder Boolean Expressions

$$Y3 = A' \times B'$$

$$Y2 = A' \times B$$

$$Y1 = A \times B'$$

$$Y0 = A \times B$$

Figure 3-1. 2-to-4 Binary Decoder Circuit

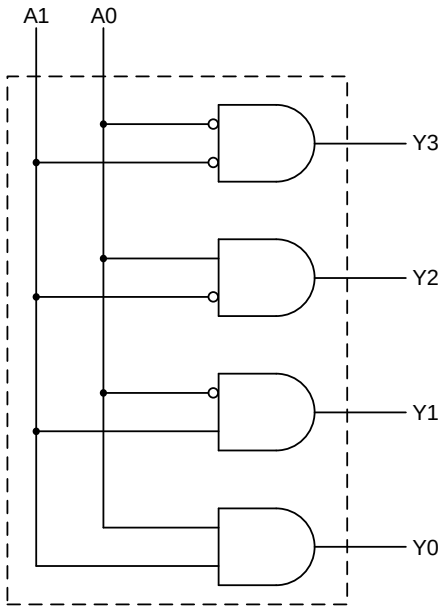


Figure 3-2. CLC1 Configuration for 2-to-4 Binary Decoder

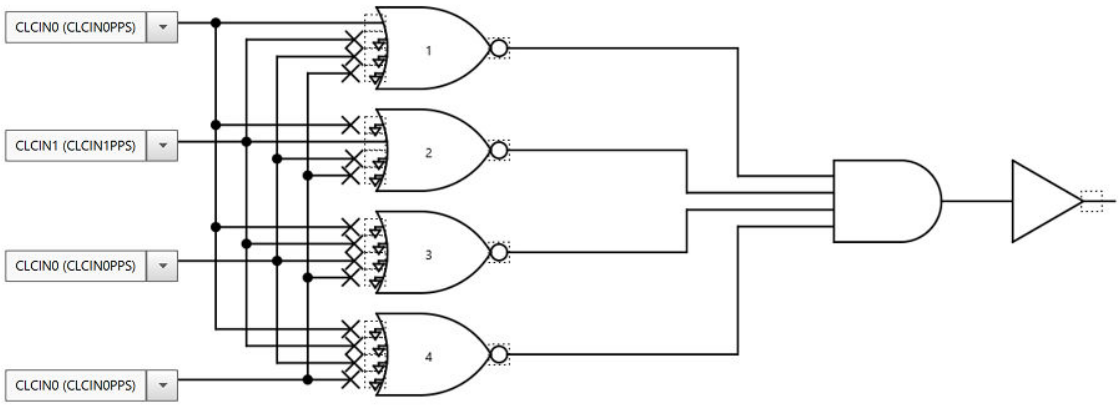


Figure 3-3. CLC2 Configuration for 2-to-4 Binary Decoder

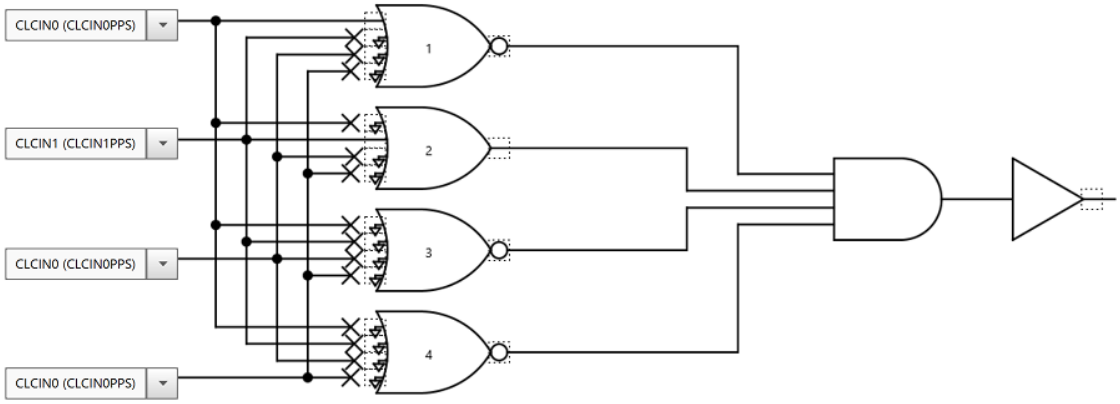


Figure 3-4. CLC5 Configuration for 2-to-4 Binary Decoder

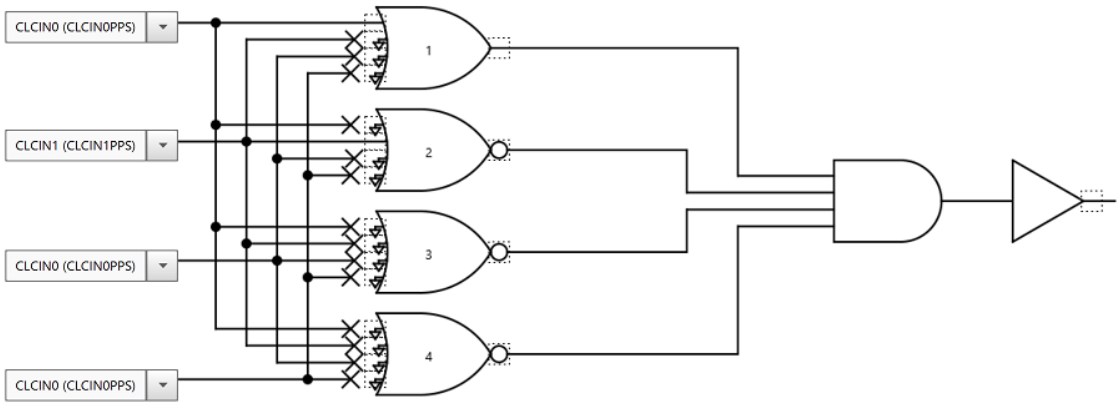
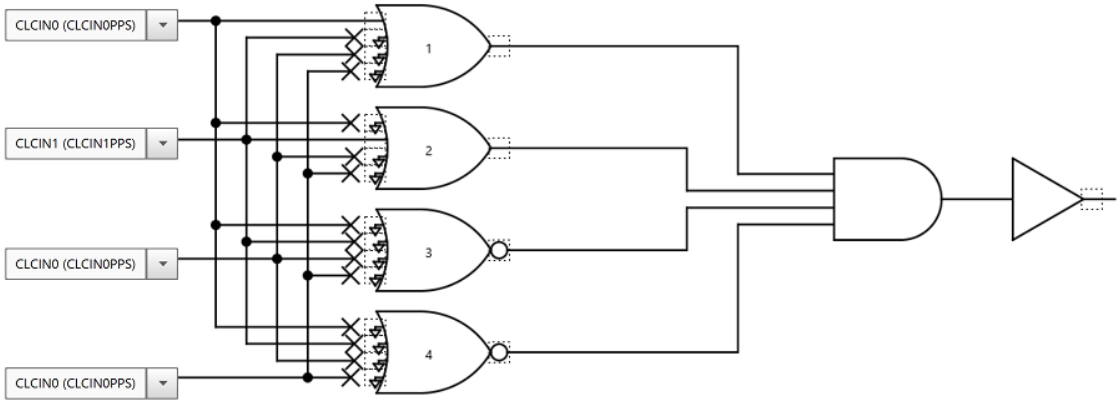


Figure 3-5. CLC6 Configuration for 2-to-4 Binary Decoder



Example 3-1. 2-to-4 Binary Decoder Initialization Code

```
/*This code block configures the CLCs
for 2-to-4 Binary Decoder.
  */

void CLC1_Initialize(void) {
    CLCSELECT = 0x00;         // SLCT 0
    CLCnPOL = 0x0F;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
    CLCnSEL2 = 0x00;          // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC1OUT 0
    CLCnCON = 0x82;           // EN enabled; INTN disabled; INTP disabled; MODE
4-input AND
}

void CLC2_Initialize(void) {
    CLCSELECT = 0x01;         // SLCT 1
    CLCnPOL = 0x0D;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
    CLCnSEL2 = 0x00;          // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC2OUT 0
    CLCnCON = 0x82;           // EN enabled; INTN disabled; INTP disabled; MODE
4-input AND
}

void CLC5_Initialize(void) {
    CLCSELECT = 0x04;         // SLCT 4
    CLCnPOL = 0x0E;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
    CLCnSEL2 = 0x00;          // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC5OUT 0
    CLCnCON = 0x82;           // EN enabled; INTN disabled; INTP disabled; MODE
4-input AND
}

void CLC6_Initialize(void) {
    CLCSELECT = 0x05;         // SLCT 5
    CLCnPOL = 0x0C;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
    CLCnSEL2 = 0x00;          // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC6OUT 0
    CLCnCON = 0x82;           // EN enabled; INTN disabled; INTP disabled; MODE
4-input AND
}
```

4. 3-to-8 Binary Decoder

In the previous example, four CLCs were used to implement a 2-to-4 binary encoder in hardware. The following example will demonstrate how to implement 3-to-8 binary decoder using the same principals. Table 4-1 below shows the truth table for the 3-to-8 binary decoder, and Figure 4-1 illustrates the resulting circuit that should be implemented using CLCs, based on the derived Boolean expressions. MCC was used to setup the CLC modules for this application, and the configuration settings can be found in the figures below. The configuration code can be found in Example 4-1.

Table 4-1. 3-to-8 Binary Decoder Truth Table

A2	A1	A0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Equation 4-1. 3-to-8 Binary Decoder Boolean Expressions

$$Y0 = A'B'C'$$

$$Y1 = A'B'C$$

$$Y2 = A'BC'$$

$$Y3 = A'BC$$

$$Y4 = AB'C'$$

$$Y5 = AB'C$$

$$Y6 = ABC'$$

$$Y7 = ABC$$

Figure 4-1. 3-to-8 Binary Decoder Circuit

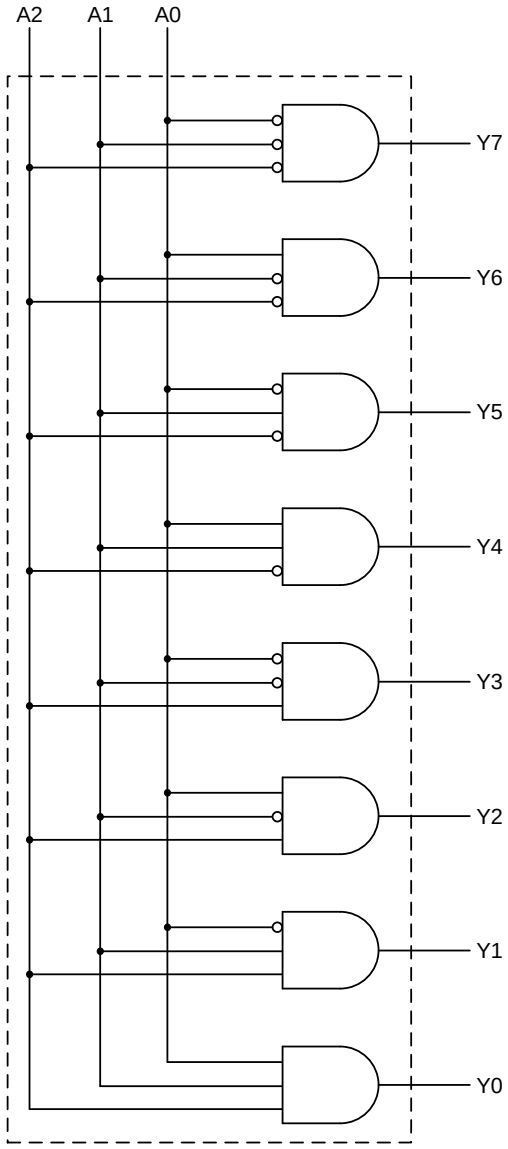


Figure 4-2. CLC1 Configuration for 3-to-8 Binary Decoder

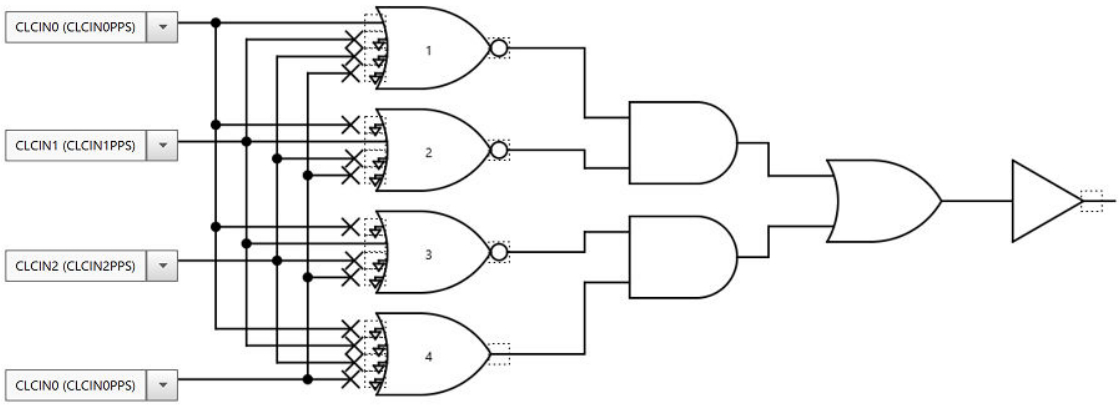


Figure 4-3. CLC2 Configuration for 3-to-8 Binary Decoder

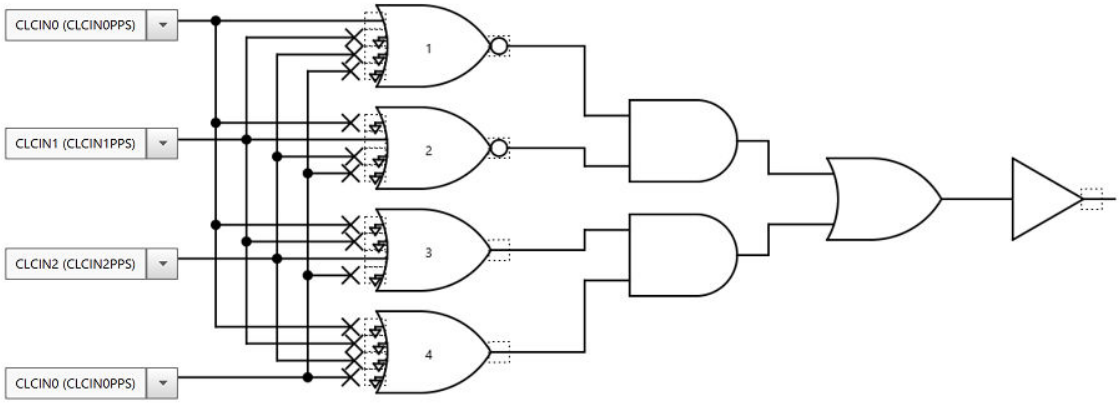


Figure 4-4. CLC3 Configuration for 3-to-8 Binary Decoder

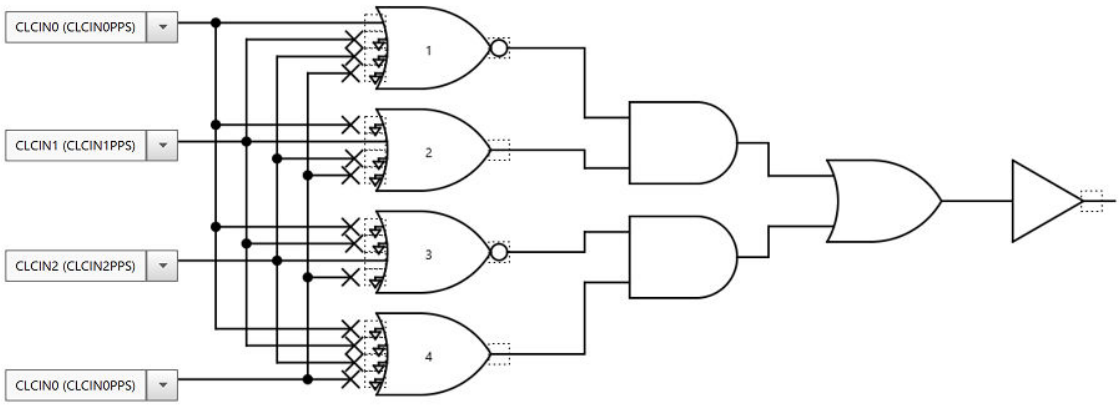


Figure 4-5. CLC4 Configuration for 3-to-8 Binary Decoder

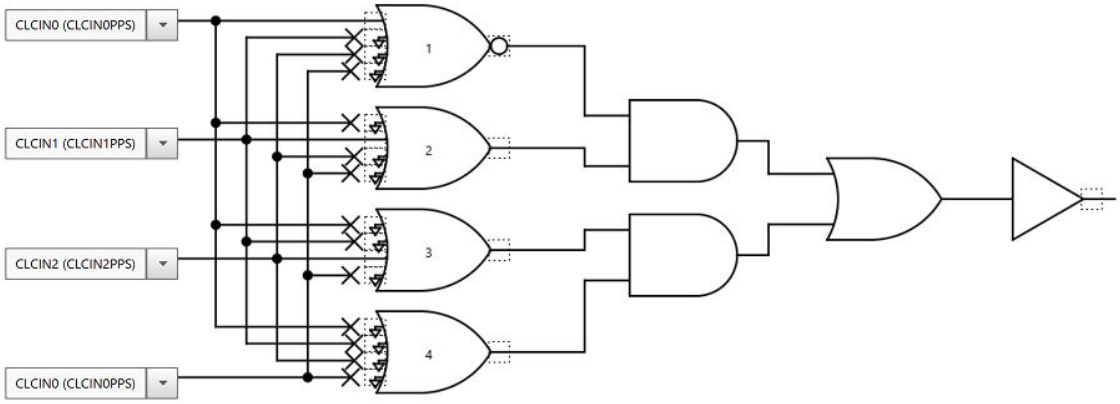


Figure 4-6. CLC5 Configuration for 3-to-8 Binary Decoder

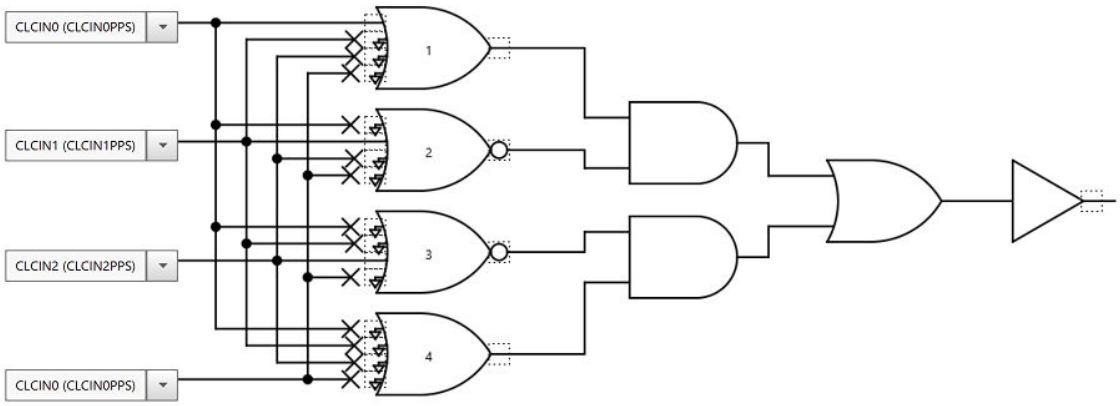


Figure 4-7. CLC6 Configuration for 3-to-8 Binary Decoder

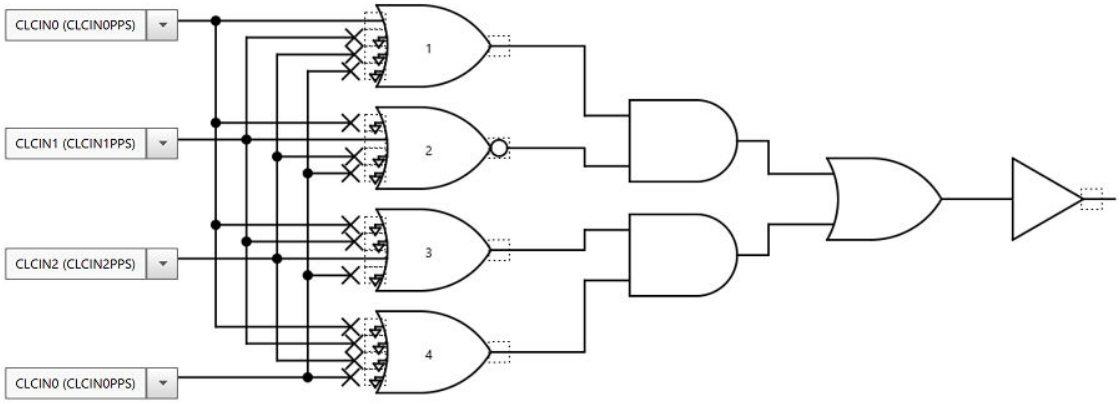


Figure 4-8. CLC7 Configuration for 3-to-8 Binary Decoder

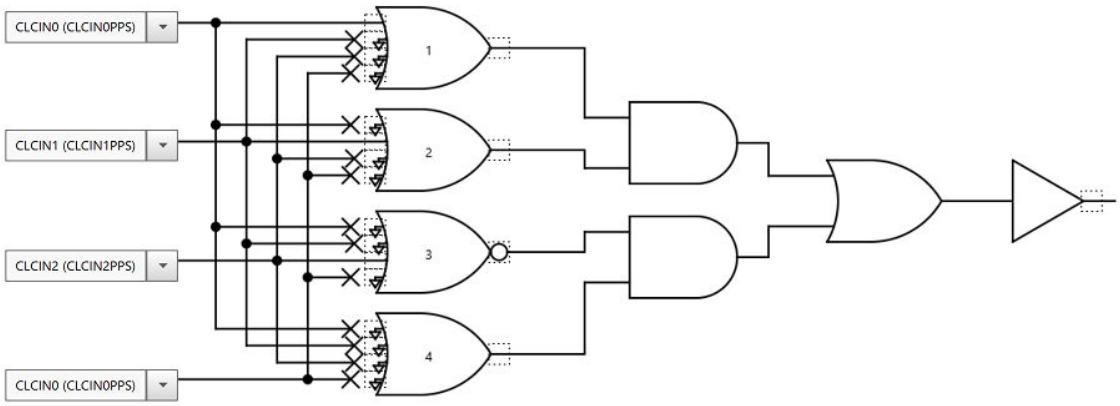
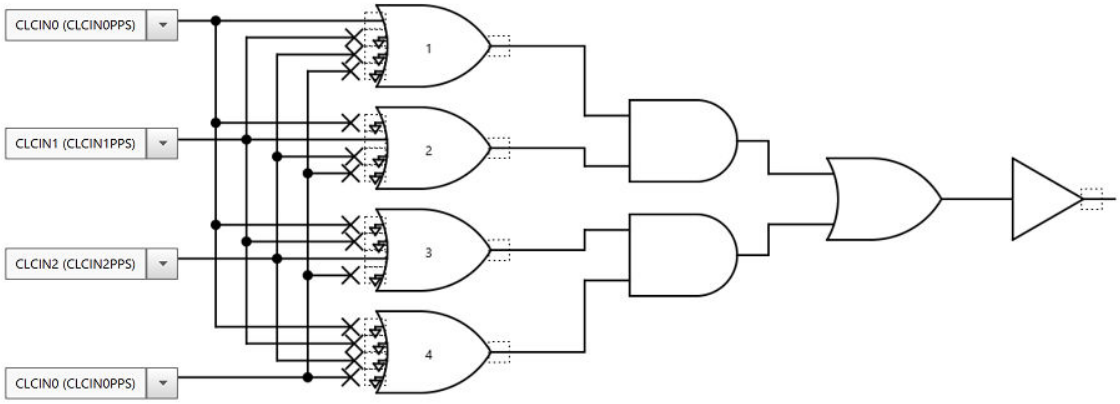


Figure 4-9. CLC8 Configuration for 3-to-8 Binary Decoder



Example 4-1. 3-to-8 Binary Decoder Initialization Code

```
/*This code block configures the CLCs
for 3-to-8 Binary Decoder.
  */

void CLC1_Initialize(void) {
    CLCSELECT = 0x00;         // SLCT 0
    CLCnPOL = 0x07;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
    CLCnSEL2 = 0x02;          // D3S CLCIN2 (CLCIN2PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x08;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC1OUT 0
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR
}

void CLC2_Initialize(void) {
    CLCSELECT = 0x01;         // SLCT 1
    CLCnPOL = 0x03;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
    CLCnSEL2 = 0x02;          // D3S CLCIN2 (CLCIN2PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;             // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x20;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC2OUT 0
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR
}

void CLC3_Initialize(void) {
    CLCSELECT = 0x02;         // SLCT 2
    CLCnPOL = 0x05;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
    CLCnSEL2 = 0x02;          // D3S CLCIN2 (CLCIN2PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x20;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC3OUT 0
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR
}

void CLC4_Initialize(void) {
    CLCSELECT = 0x03;         // SLCT 3
    CLCnPOL = 0x01;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
    CLCnSEL2 = 0x02;          // D3S CLCIN2 (CLCIN2PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x20;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC4OUT 0
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR
}

void CLC5_Initialize(void) {
    CLCSELECT = 0x04;         // SLCT 4
    CLCnPOL = 0x06;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
```

```
    CLCnSEL2 = 0x02;          // D3S CLCIN2 (CLCIN2PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x20;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC5OUT 0
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR
}

void CLC6_Initialize(void) {
    CLCSELECT = 0x05;         // SLCT 5
    CLCnPOL = 0x02;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS)
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS)
    CLCnSEL2 = 0x02;          // D3S CLCIN2 (CLCIN2PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x20;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC6OUT 0
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR
}

void CLC7_Initialize(void) {
    CLCSELECT = 0x06;         // SLCT 6;
    CLCnPOL = 0x04;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS);
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS);
    CLCnSEL2 = 0x02;          // D3S CLCIN2 (CLCIN2PPS);
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS);
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x20;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC7OUT 0;
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR;
}

void CLC8_Initialize(void) {
    CLCSELECT = 0x07;         // SLCT 7;
    CLCnPOL = 0x00;              // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x00;          // D1S CLCIN0 (CLCIN0PPS);
    CLCnSEL1 = 0x01;          // D2S CLCIN1 (CLCIN1PPS);
    CLCnSEL2 = 0x02;          // D3S CLCIN2 (CLCIN2PPS);
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS);
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x20;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC8OUT 0;
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR;
}
```

5. Edge Detection

Edge detection is useful in cases when a system needs to respond to a change of state on a signal. Edge detection circuits can be implemented in hardware using the CLC module, and operate independent of the core. This allows the CPU core to tend to other important tasks, and eliminates any delays that would be involved in processing these signals using the CPU without CLC modules. This application of the CLC module can be very useful in cases where the CPU core is in Sleep mode to save power. The CLC module will remain active in Sleep, and the circuit that was configured will continue to capture the input edges as designed. The circuit implemented in this application is capable of detecting any rising or falling edge of the input signal. In this example the MFINTOSC signal is used as the input to the edge detection circuit.

Figure 5-1. CLC1 Configuration for D Flip-Flop

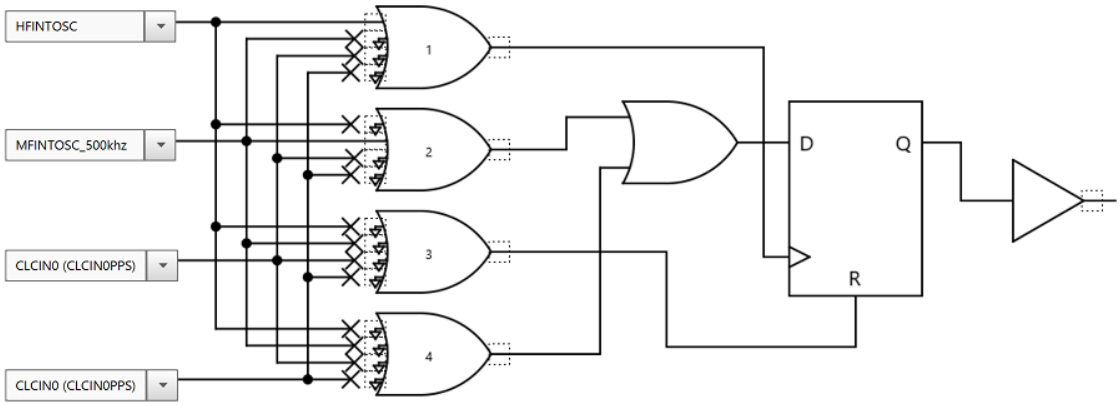


Figure 5-2. CLC2 Configuration for Rising Edge Detection

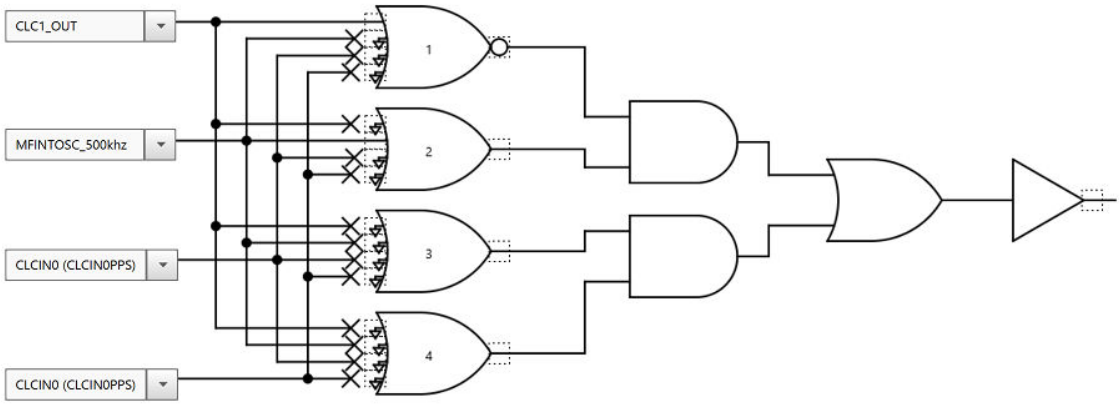


Figure 5-3. CLC3 Configuration for Falling Edge Detection

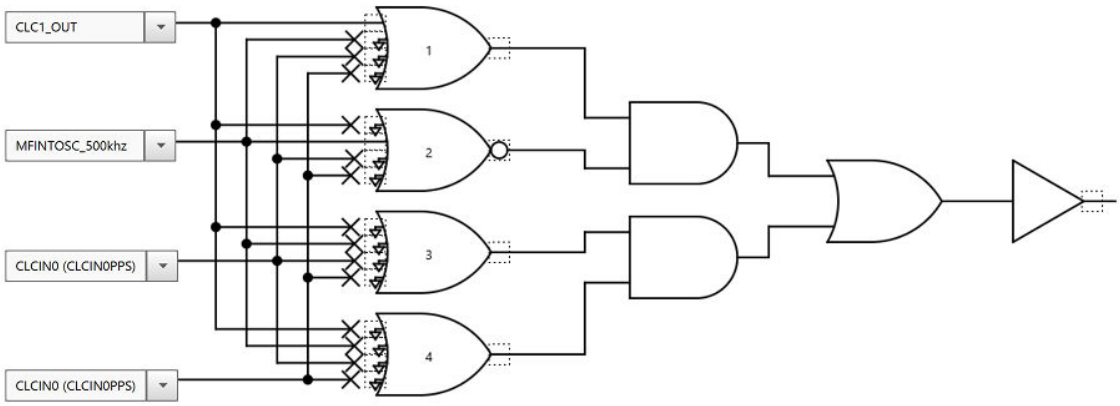


Figure 5-4. CLC4 Configuration for any Rising or Falling Edge Detection

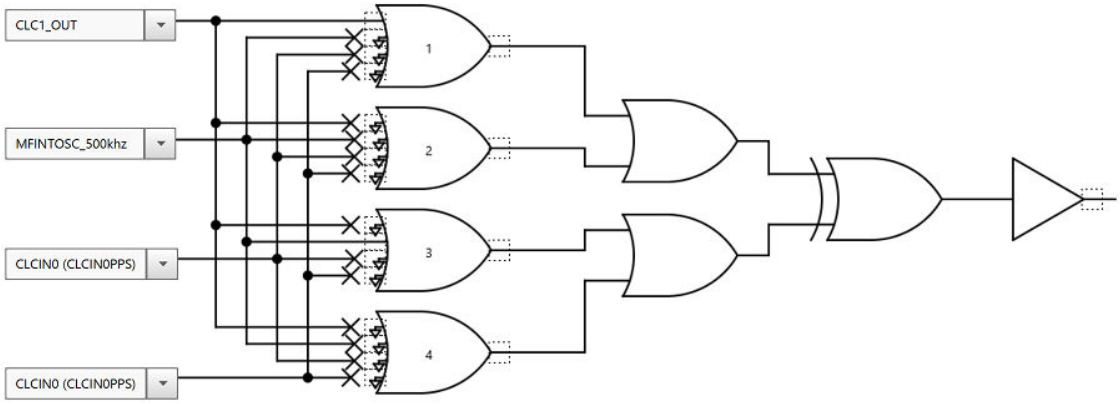
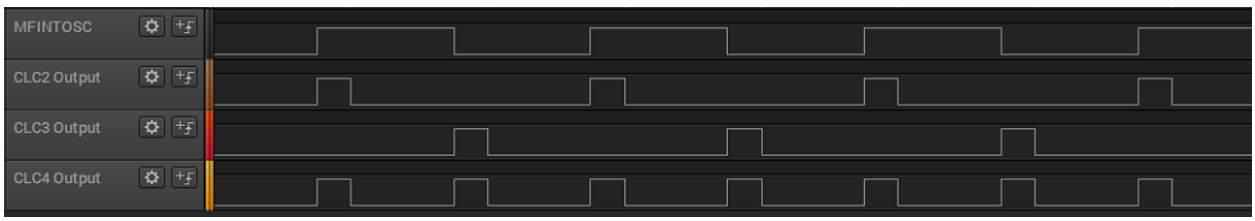


Figure 5-5. Output Signals



Example 5-1. Edge Detection Initialization Code

```
/*This code block configures the CLCs
for Edge Detection.
  */

void CLC1_Initialize(void) {
    CLCSELECT = 0x00;         // SLCT 0
    CLCnPOL = 0x00;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x09;          // D1S HFINTOSC
    CLCnSEL1 = 0x0B;          // D2S MFINTOSC_500khz
    CLCnSEL2 = 0x00;          // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC1OUT 0
    CLCnCON = 0x85;           // EN enabled; INTN disabled; INTP disabled; MODE
2-input D flip-flop with R
}

void CLC2_Initialize(void) {
    CLCSELECT = 0x01;         // SLCT 1
    CLCnPOL = 0x01;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x33;          // D1S CLC1_OUT
    CLCnSEL1 = 0x0B;          // D2S MFINTOSC_500khz
    CLCnSEL2 = 0x00;          // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC2OUT 0
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR
}

void CLC3_Initialize(void) {
    CLCSELECT = 0x02;         // SLCT 2
    CLCnPOL = 0x02;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x33;          // D1S CLC1_OUT
    CLCnSEL1 = 0x0B;          // D2S MFINTOSC_500khz
    CLCnSEL2 = 0x00;          // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC3OUT 0
    CLCnCON = 0x80;           // EN enabled; INTN disabled; INTP disabled; MODE
AND-OR
}

void CLC4_Initialize(void) {
    CLCSELECT = 0x03;         // SLCT 3
    CLCnPOL = 0x00;           // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x33;          // D1S CLC1_OUT
    CLCnSEL1 = 0x0B;          // D2S MFINTOSC_500khz
    CLCnSEL2 = 0x00;          // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;          // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;          // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x00;          // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x08;          // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;          // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;           // CLC4OUT 0
    CLCnCON = 0x81;           // EN enabled; INTN disabled; INTP disabled; MODE
OR-XOR
}
```

6. Pseudo Random Number Generator Using the SPI Module

This example implements three CLCs for logic/flip-flops as well as the SPI module as a shift register. Two CLCs are configured as D flip-flops with both S and R tied to '0', and the third is configured as an XOR gate. Both D flip-flops use the SCK signal from the SPI module as their clock, and the first flip-flop uses the SDO output of the SPI module as its D input. The second flip-flop uses the Q output of the first flip-flop as its D input. The Q outputs of both flip-flops are then XOR'ed together and fed back to the SPI module as the SDI input. This circuit works by feeding an initial 8-bit seed value into the SPI transmit FIFO, which the SPI will then send out onto the SDO signal. The CLC logic will then create a new byte (bit-by-bit) and input that data back into the SPI module using the SDI input. Doing this uses the original seeded value that was transmitted by the SPI module to create a new random value that is transferred to the RXFIFO of the SPI module all in hardware. The SPI module operates in Full-Duplex mode, meaning that it will transfer data whenever the TXFIFO has been written and if the RXFIFO is not already full. Lastly, the DMA module must be configured so that it transfers data from the RXFIFO to the TXFIFO, which will allow the circuit to continuously generate psuedo-random numbers. The configuration code for DMA and PPS settings can be found in Example 6-2.

Figure 6-1. Pseudo Random Number Generator Circuit

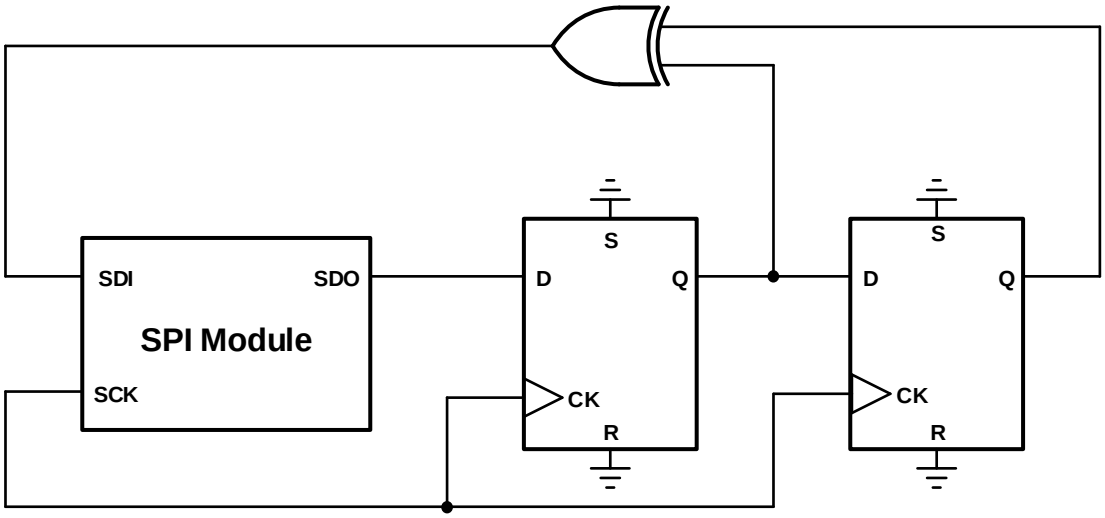


Figure 6-2. CLC1 Configuration

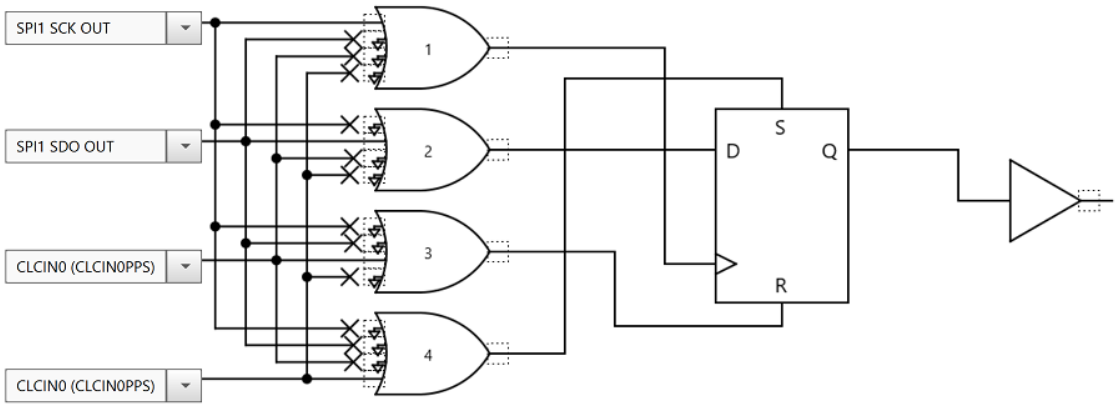


Figure 6-3. CLC2 Configuration

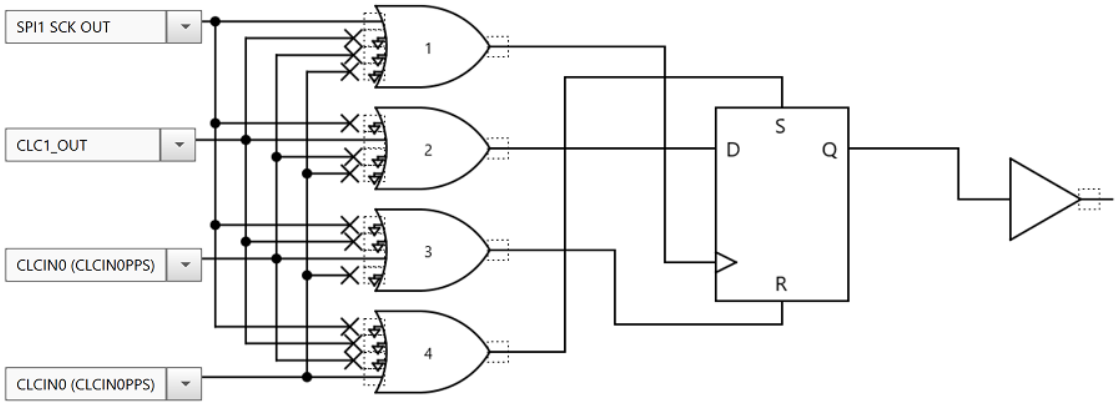
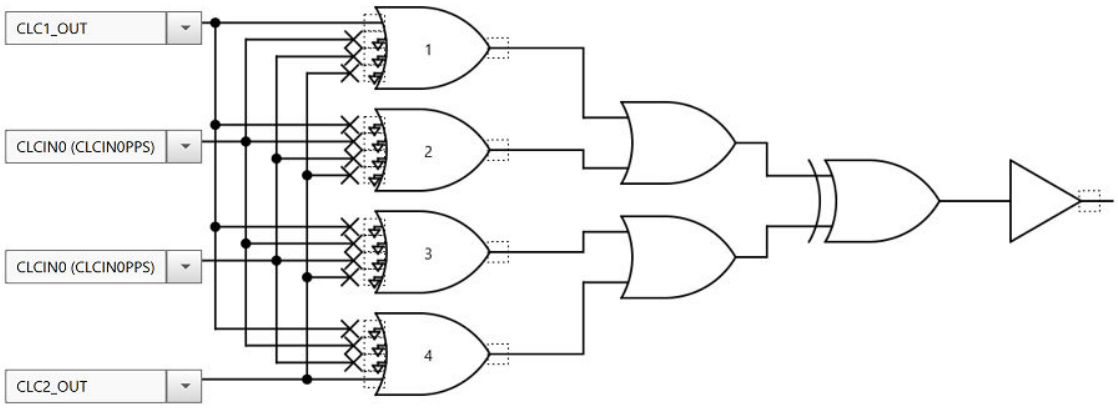


Figure 6-4. CLC3 Configuration



Example 6-1. Pseudo Random Number Generator Initialization Code

```
/*This code block configures the CLCs
for Pseudo Random Number Generator.
  */

void CLC1_Initialize(void) {
    CLCSELECT = 0x00;           // SLCT 0
    CLCnPOL = 0x00;          // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x41;            // D1S SPI1 SCK OUT
    CLCnSEL1 = 0x40;            // D2S SPI1 SDO OUT
    CLCnSEL2 = 0x00;            // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;            // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;            // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;            // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x20;            // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x40;            // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;             // CLC1OUT 0
    CLCnCON = 0x84;             // EN enabled; INTN disabled; INTP disabled;
MODE 1-input D flip-flop with S and R
}

void CLC2_Initialize(void) {
    CLCSELECT = 0x01;           // SLCT 1
    CLCnPOL = 0x00;             // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x41;            // D1S SPI1 SCK OUT
    CLCnSEL1 = 0x33;            // D2S CLC1_OUT
    CLCnSEL2 = 0x02;            // D3S CLCIN2 (CLCIN2PPS)
    CLCnSEL3 = 0x00;            // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;            // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;            // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x20;            // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x40;            // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;             // CLC2OUT 0;
    CLCnCON = 0x84;             // EN enabled; INTN disabled; INTP disabled;
MODE 1-input D flip-flop with S and R;
}

void CLC3_Initialize(void) {
    CLCSELECT = 0x02;           // SLCT 2
    CLCnPOL = 0x00;             // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x33;            // D1S CLC1_OUT;
    CLCnSEL1 = 0x00;            // D2S CLCIN0 (CLCIN0PPS);
    CLCnSEL2 = 0x00;            // D3S CLCIN0 (CLCIN0PPS);
    CLCnSEL3 = 0x34;            // D4S CLC2_OUT;
    CLCnGLS0 = 0x02;            // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;            // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;            // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x40;            // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;             // CLC3OUT 0;
    CLCnCON = 0x81;             // EN enabled; INTN disabled; INTP disabled;
MODE OR-XOR;
}
```

Example 6-2. DMA Initialization Code

```
/*This code block configures the DMA
and PPS Settings for Pseudo Random Number Generator.
  */

void DMA1_Initialize(void) {
    DMASELECT = 0x00;           // Select DMA1
    DMAnSSA = &SPI1RXB;         // Source is SPI1 RX Buffer
    DMAnDSA = &SPI1TXB;         // Destination is SPI1 TX Buffer
    DMAnCON1 = 0x0B;            // DMODE unchanged, SMODE increments, SIRQEN
cleared upon reload
    DMAnSSZ = 0x01;             // Source size is one (one buffer to write)
    DMAnDSZ = 0x01;             // Destination size is one (one buffer to
write)
    DMAnSIRQ = 0x18;            // DMA Transfer Trigger Source = SPI1RXIF
    DMAnAIRQ = 0x14;            // DMA Transfer Abort Source = DMA1SCNTIF
```

```
    PIR2bits.DMA1SCNTIF = 0;    // Clear Source Count Interrupt Flag bit
    PIR2bits.DMA1AIF = 0;       // Clear abort Interrupt Flag bit
    PIE2bits.DMA1AIE = 1;       // Enable abort Interrupt

    asm("BCF INTCON0,7");       // Lock the prority
    asm ("BANKSEL PRLOCK");
    asm ("MOVLW 0x55");
    asm ("MOVWF PRLOCK");
    asm ("MOVLW 0xAA");
    asm ("MOVWF PRLOCK");
    asm ("BSF PRLOCK, 0");
    asm("BSF INTCON0,7");
}

// PPS Settings
    RC3PPS = 0x00;              // RC3->SPI1:SCK1
    CLCIN0PPS = 0x10;           // RC0->CLC3:CLCIN0;  Must connect to GND
    RB0PPS = 0x03;              // RB0->CLC3:CLC3_out
    SPI1SDIPPS = 0x08;          // RB0->SPI1:SDI1;  Connects to CLC3_out

    SPI1TXB = 0x2C;             // Load SPI1TXB with beginning 'seed' value
```

7. Quadrature Clock Generator

The CLC module can be used to create a quadrature clock generator which consists of two output clocks 90° out of phase of one another. A quadrature clock generator can be useful in many motor control and signal processing applications. For this application, two CLC modules in D Flip-Flop mode are used to create the circuit illustrated in Figure 7-1. The output signals of this circuit are half the frequency of the input signal. Figure 7-2 and Figure 7-3 show the configuration settings of each CLC module used in MCC. Example 7-1 shows the configuration code used to setup the modules for this application.

Two CLCs in D Flip-Flop mode are used to create the following circuit. The output signals are half the frequency of the input signal.

Figure 7-1. Quadrature Clock Generator Logic Diagram

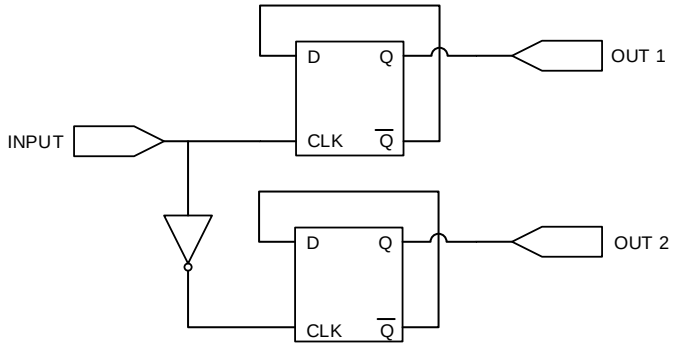


Figure 7-2. CLC1 Configuration

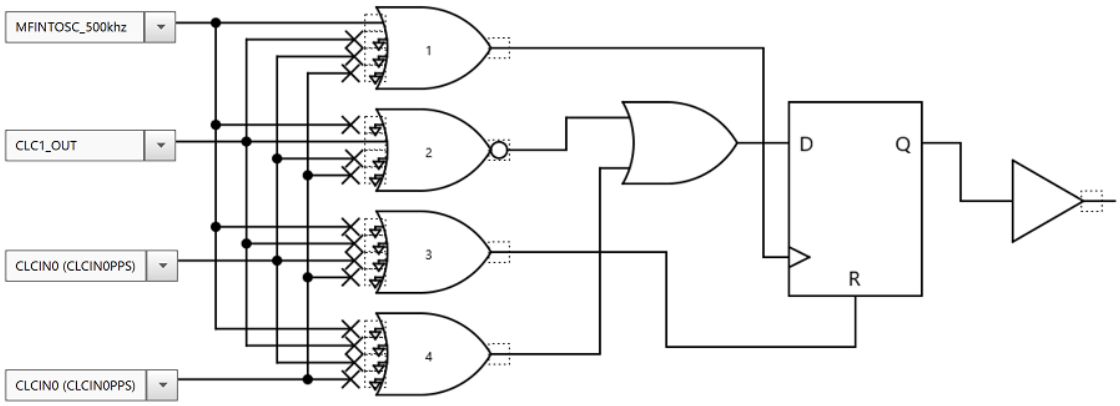
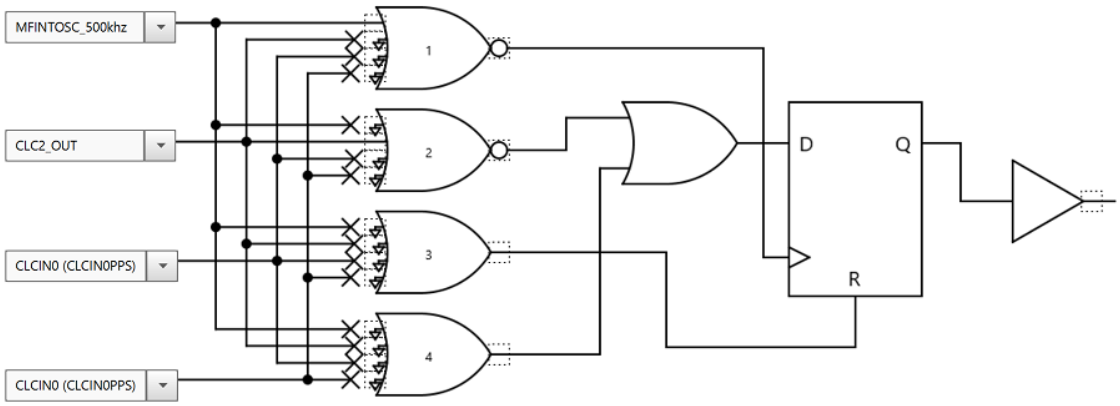


Figure 7-3. CLC2 Configuration



Example 7-1. Quadrature Clock Generator Initialization Code

```
/*This code block configures the CLCs
for Quadrature Clock Generator.
  */

void CLC1_Initialize(void) {
    CLCSELECT = 0x00;     // SLCT 0
    CLCnPOL = 0x02;       // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x0B;      // D1S MFINTOSC_500khz
    CLCnSEL1 = 0x33;      // D2S CLC1_OUT
    CLCnSEL2 = 0x00;      // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;      // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;      // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08;      // CLCn Gate 2 Logic Selection
    CLCnGLS2 = 0x00;      // CLCn Gate 3 Logic Selection
    CLCnGLS3 = 0x00;      // CLCn Gate 4 Logic Selection
    CLCDATA = 0x00;       // CLC1OUT 0
    CLCnCON = 0x85;       // EN enabled; INTN disabled; INTP disabled; MODE 2-
input D flip-flop with R
}

void CLC2_Initialize(void) {
    CLCSELECT = 0x01;     // SLCT 1
    CLCnPOL = 0x03;       // Gate and CLCnOUT Output polarity Selection
    CLCnSEL0 = 0x0B;      // D1S MFINTOSC_500khz
    CLCnSEL1 = 0x34;      // D2S CLC2_OUT
    CLCnSEL2 = 0x00;      // D3S CLCIN0 (CLCIN0PPS)
    CLCnSEL3 = 0x00;      // D4S CLCIN0 (CLCIN0PPS)
    CLCnGLS0 = 0x02;      // CLCn Gate 1 Logic Selection
    CLCnGLS1 = 0x08       // CLCn Gate 1 Logic Selection
    CLCnGLS2 = 0x00       // CLCn Gate 1 Logic Selection
    CLCnGLS3 = 0x00;      // CLCn Gate 1 Logic Selection
    CLCDATA = 0x00;       // CLC2OUT 0
    CLCnCON = 0x85;       // EN enabled; INTN disabled; INTP disabled; MODE 2-
input D flip-flop with R
}
```

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with

Trademarks

ISBN: 978-1-5224-5973-6

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820