10 Sequential logic circuits – applications based on counters

10.1 Objectives

Several methods are presented for extending the counting loop, by cascading smaller counters with reduced number of bits. The differences between binary and decade counters are analyzed, when implemented with multiple counters. The implementation of *modulo p* counters is studied, while highlighting the particularities specific to counters used.

10.2 Theoretical considerations

Digital systems with larger number of bits incorporate counters of higher capacity. The size of the counters can be extended by cascading smaller counters. For instance, using the 4-bit counters studied in the previous work, the size can be extended to any multiple of 4 bits by connecting several such units. Consequently, the counting loop is extended from 2^4 values to $2^{n\text{x}4}$, where *n* represents the number of cascaded counters.

Another situation frequently met in digital systems is the reduction of the counting loop to a subset of values from the total of 2^n *n*-bit values. A counter of capacity *p*, where $p<2^n$, is called a *modulo p* counter. The *p* values can be randomly chosen, meaning there can be several counting loops defined for the same *p*. For instance, the decade counter representing the decimal digits (the counting loop 0÷9) is *modulo 10*.

10.2.1 Extending the counting loop

The increase of the counting loop, using cascaded counters, respects the following principle: the counting for the lowest ranking counter is enabled every clock cycle; any other counter from the list of cascaded counters will activate its counting once, for each end of the inferior counter counting loop.

When cascading 74193 up-down binary counters, the *carry* and *borrow* flags must be connected to the clock signals of the superior rank counter (if any), as highlighted in Figure 10. 1:

- At count up, the counter on $Q_{3:0}$ has its CountUp input connected to the clock signal, while the CountDown input is connected to 1. Consequently, the $\overline{\text{Borrow}}$ output flag will be kept inactive to 1, meaning the $Q_{7:4}$ counter will increment every rising edge registered on the $\overline{\text{Carry}}$ flag, specifically on the transition from 15 to 0 of the $Q_{3:0}$ counter (Figure 10. 2 – left).
- At count down, the $Q_{3:0}$ counter connects its CountDown input to the clock signal, and CountUp=1. Hence, the $\overline{\text{Carry}}$ flag will be 1 (inactive), and the rising edge on the $\overline{\text{Borrow}}$ flag, at the 0 to 15 transition (Figure 10. 2 – right), will decrement the $Q_{7:4}$ counter.

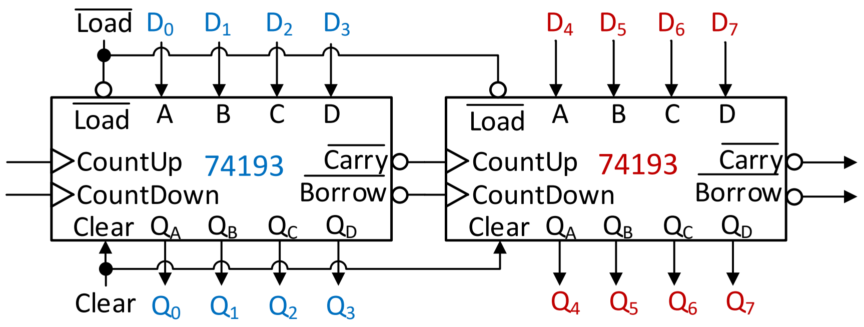


Figure 10. 1 Cascading two 74193 counters

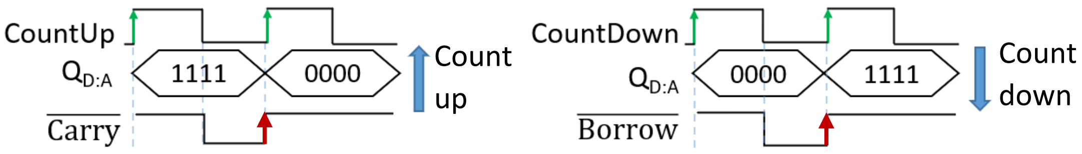


Figure 10. 2 The counter 74193 generates a rising edge on $\overline{\text{Carry}}$ or $\overline{\text{Borrow}}$, at the end of the loop

Note: The 8-bit loop on the outputs has 256 values, in range 0÷255 (Figure 10. 4 – left).

When cascading up-down decade counters 74192, their interconnection follow similar rules: the output flags $\overline{TC}_U$ and $\overline{TC}_D$ are connected to clock inputs CP_U and CP_D, respectively, as in Figure 10. 3. The output values must be interpreted in base 10, after converting 4-bit pairs to corresponding decimal digits. At runtime, each 10 consecutive values on $Q_{3:0}$ correspond to a change on $Q_{7:4}$. The values registered on outputs $Q_{7:0}$ generate the loop 00÷99: 00, 01, 02, … 09, 10, 11, 12, … 19, 20, 21, 22, … 99, 00, 01, 02, … (Figure 10. 4 – right).

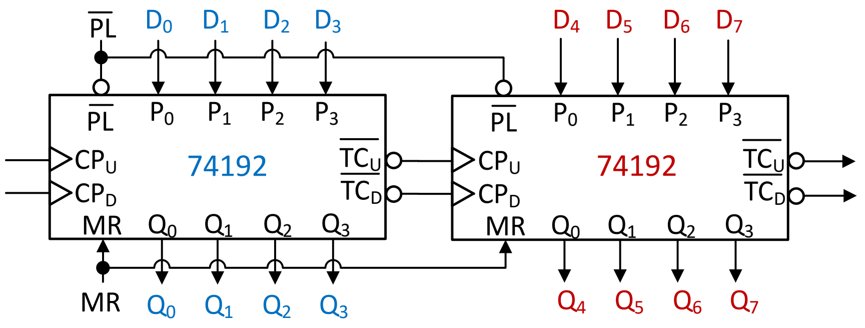
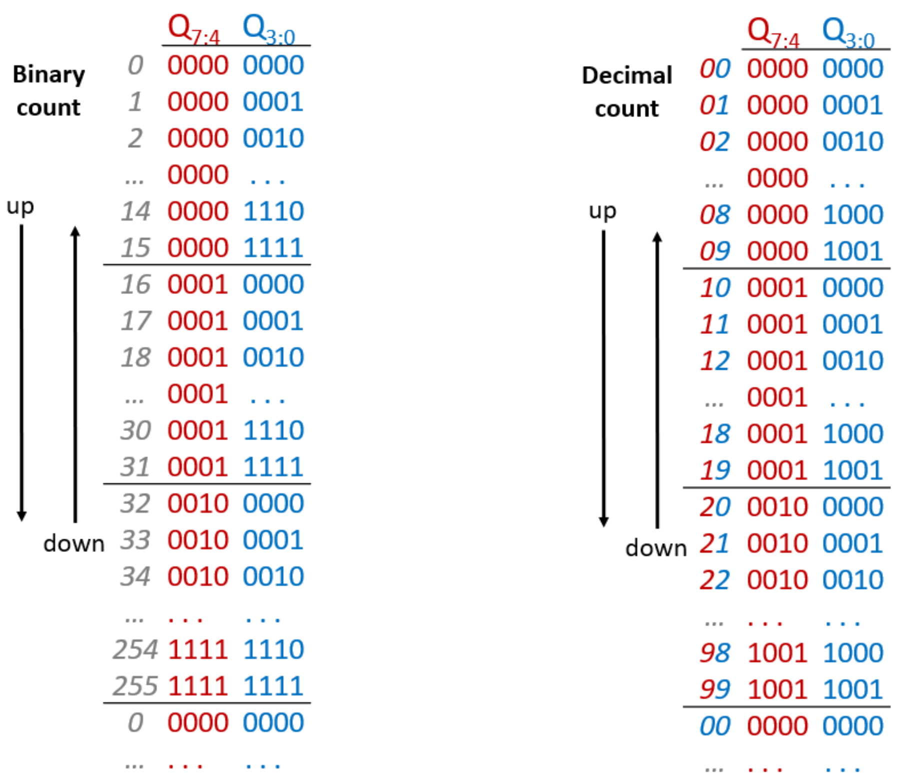


Figure 10. 3 Cascading two 74192 counters

Note: The asynchronous reset and the asynchronous load with $D_{7:0}$ are shared by all counters. Their effect is simultaneous.



Binary count (up ↓, down ↑)

	$Q_{7:4}$	$Q_{3:0}$
0	0000	0000
1	0000	0001
2	0000	0010
…	0000	. . .
14	0000	1110
15	0000	1111
16	0001	0000
17	0001	0001
18	0001	0010
…	0001	. . .
30	0001	1110
31	0001	1111
32	0010	0000
33	0010	0001
34	0010	0010
…	. . .	. . .
254	1111	1110
255	1111	1111
0	0000	0000
…	. . .	. . .

Decimal count (up ↓, down ↑)

	$Q_{7:4}$	$Q_{3:0}$
00	0000	0000
01	0000	0001
02	0000	0010
…	0000	. . .
08	0000	1000
09	0000	1001
10	0001	0000
11	0001	0001
12	0001	0010
…	0001	. . .
18	0001	1000
19	0001	1001
20	0010	0000
21	0010	0001
22	0010	0010
…	. . .	. . .
98	1001	1000
99	1001	1001
00	0000	0000
…	. . .	. . .

Figure 10. 4 The 8-bit counting loops: binary (left) and decimal (right)

It is also possible to cascade binary and decade counters. If the objective is a binary counter, the less significant bits $Q_{3:0}$ should be the outputs of 74193, and the rest of bits $Q_{7:0}$, should be the outputs of 74192 (Figure 10. 5). The ensemble will generate a counting loop in range 0÷159 (159_{10}=1001 1111_2). Otherwise, if the decimal counter 74192 counts on $Q_{3:0}$ the loop will be discontinuous, because after 10 consecutive values, the following 7 values will be skipped: 0, 1, … 9, 16, 17, 18, … 25, 32 … .

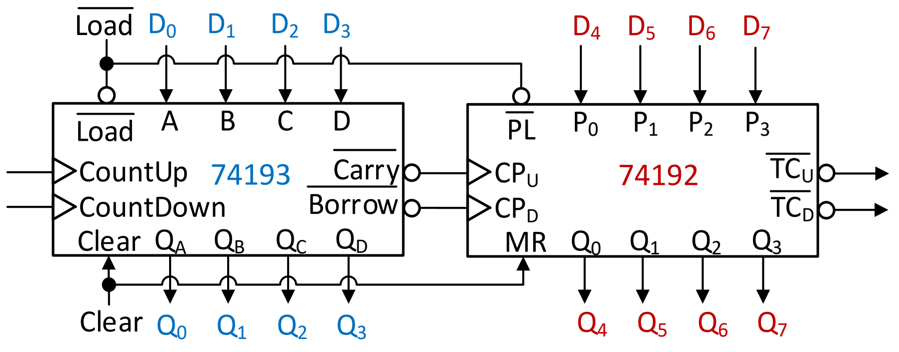


Figure 10. 5 Cascading counters 74192 and 74193 to generate a binary counter

When cascading 74163 count-up binary counters (Figure 10. 6 – left), they must share a common clock signal. The counter on $Q_{3:0}$ should always be active, by enabling inputs ENP=ENT=1. The rest of counters should have ENP=1, and ENT should be connected to the RCO output flag of the inferior counter. Since RCO activates when $Q_{3:0}$=1111, the higher rank counter increments on the $Q_{3:0}$ transition to 0000, as shown in the timing diagram from Figure 10. 6. Afterwards, the RCO flag is disabled for the next 15 cycles. The resulting 8-bit binary loop is 0÷255.

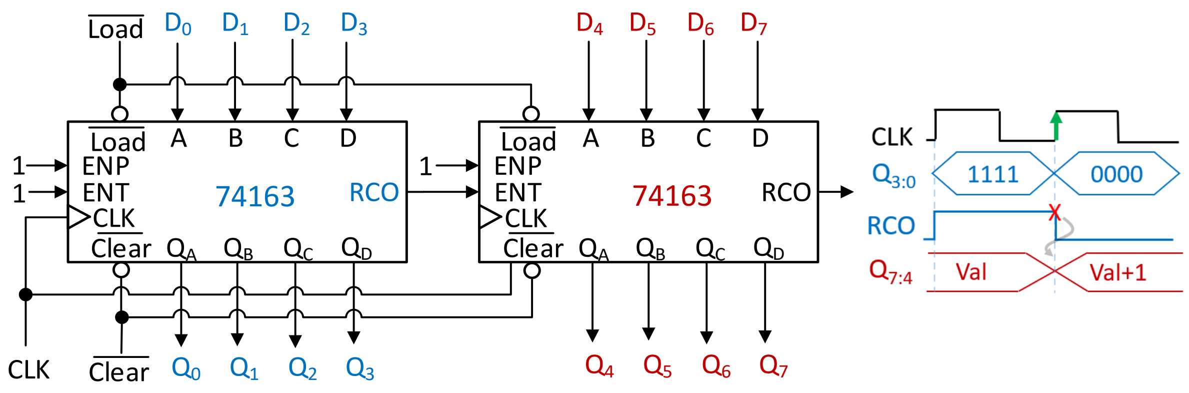


Figure 10. 6 Cascading two 74163 counters (left); the counting for the superior rank counter is enabled when the inferior rank counter ends its loop (right)

Cascading decade count-up counters 74162 (Figure 10. 7) follows the same principles as for 74163. The resulting 8-bit counting loop has 2 decimal digits, in range 00÷99.

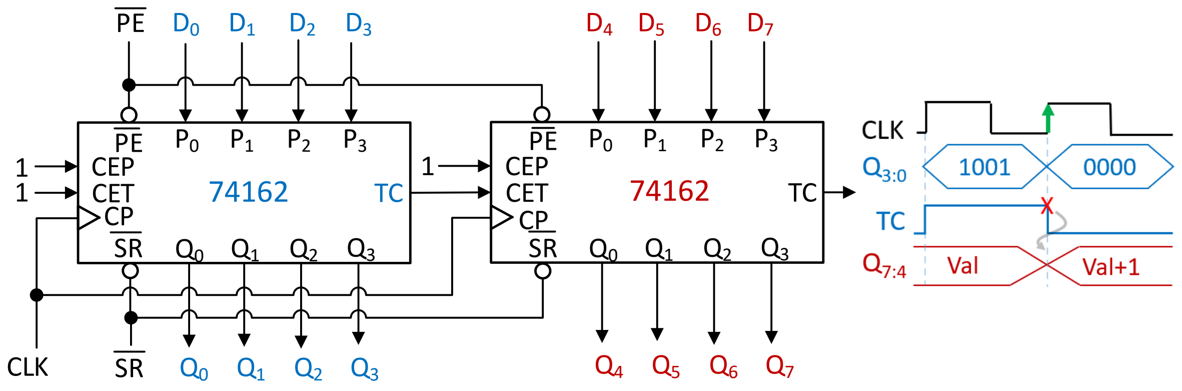


Figure 10. 7 Cascading two 74162 counters (left); the counting for the superior rank counter is enabled when the inferior rank counter ends its loop (right)

Figure 10. 8 highlights an 8-bit counter implemented using a decade counter 74162, and a binary counter 74163. To avoid loop discontinuities, the binary counter must be placed on outputs $Q_{3:0}$.

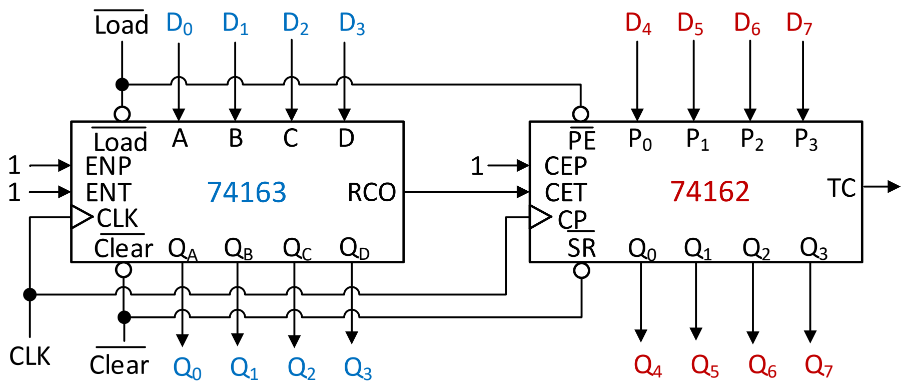


Figure 10. 8 Cascading counters 74162 and 74163 to generate a binary counter

The binary loop of the 8-bit counter in Figure 10. 8 is in range 0÷159 (159_{10}=$1001\ 1111_2$).

10.2.2 *Modulo p* counters with continuous loop

The loop of a *modulo p* counter contains *p n*-bit values, out of the 2^n possible values. Continuous loops contain consecutive values, but the start value, the end value and the counting direction can be customized. Considering the output bits of the counter are linked to other synchronous units, only values registered (visible) at the rising edge of the clock signal are important. Other intermediate values are ignored; hence they are not considered part of the counting loop.

The strategy behind implementing *modulo p* counters using 74192, 74193, 74162 and 74163 units, is to load the start value after reaching the end value. If the start value is 0, the load can be replaced by reset. The end value can be decoded with additional logic, or using the *carry*/*borrow* flags, when possible. **Note**: The counters used when implementing *modulo p* counters should be able to count all the values in the loop. For instance, when using a decade counter, it is possible to implement only *modulo p* counters with values in range 0÷9.

Note: If the counter is 0ed at startup, and this value is outside the *modulo p* loop, several clock cycles will be required before entering the counting loop.

When using 74192 and 74193 counters, the reset and load commands are asynchronous. If *Val* represents the end of loop, a reset or load will overwrite *Val* immediately, before the next rising edge. Consequently, *Val* is removed from the loop. To avoid such effect, the solution is to delay the load for one clock cycle, therefore, to detect *Val*+1 at count up or *Val*-1 at count down. Considering the start of loop is $D_{3:0}$, the implementation is highlighted in Figure 10. 9. Inputs A, B, C, D will be connected to 0 (GND) or 1 (VCC) as indicated by $D_{3:0}$. If $D_{3:0}$=7 (7_{10}=0111_2), then A=1, B=1, C=1 and D=0, respectively. **Note**: When not used, Clear must be disabled. The values of the signals when the loop is restarted, are highlighted in Figure 10. 10.

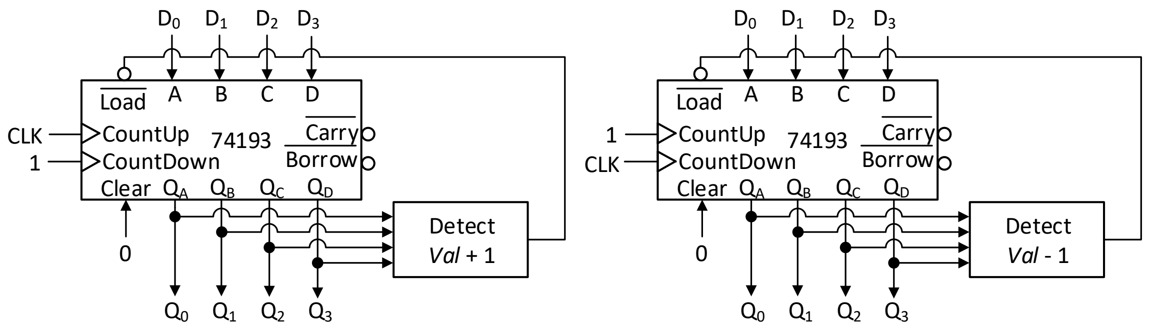
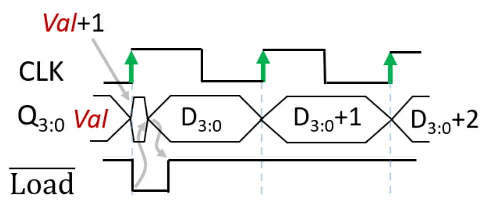


Figure 10. 9 Implementing loop $D_{3:0}$÷*Val* using counter 74193; count up loop (left) and count down loop (right)



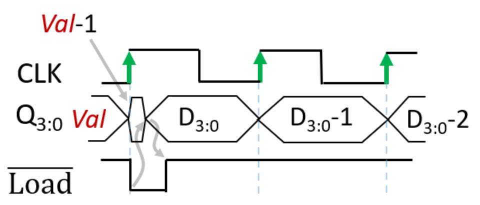
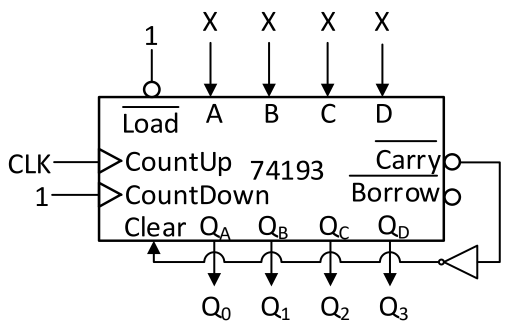


Figure 10. 10 Signal values at count up (left) and count down (right) when restarting the loop $D_{3:0}$÷*Val*, implemented with 74193

The presence of a certain value on the outputs of the counter can be detected by implementing the corresponding minterm, using an AND gate, but knowing that $\overline{\text{Load}}$ is active low, the NAND gate is more appropriate. If the decoded value is 15 or 0, the $\overline{\text{Carry}}$ and $\overline{\text{Borrow}}$ flags can be used instead. For instance, when implementing a 0÷14 loop counter, value 15 can be decoded using the $\overline{\text{Carry}}$ flag. Loading the start value 0 can be issued by enabling Clear and keeping $\overline{\text{Load}}$ inactive (Figure 10. 11 – left). When implementing loop 7÷15, the start value 7 should be loaded when detecting 0 (0=15+1), as in Figure 10. 11 – right.



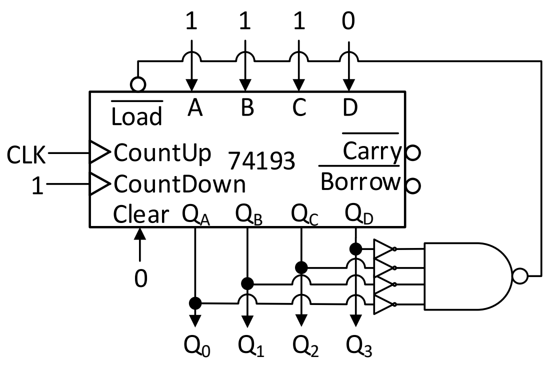
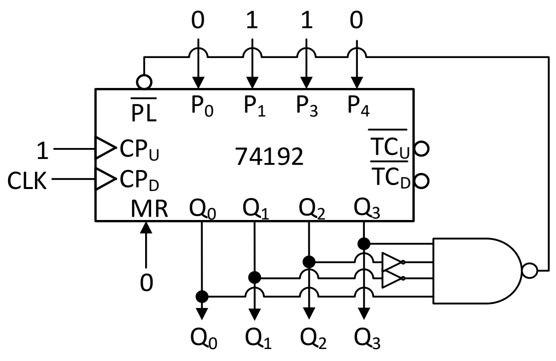


Figure 10. 11 Loops 0÷14 (left) and 7÷15 (right), implemented using counter 74193

The implementation using the decade counter 74192 is similar, excepting the counting loop, which must in range 0÷9. Figure 10. 12 highlights the implementation of loops 6÷0 (6 is loaded when 9 is detected) and 8÷1 (8 is loaded when 0 is detected).



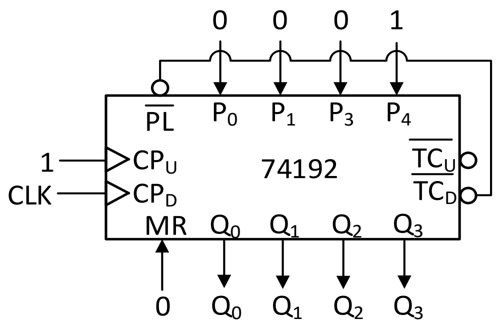


Figure 10. 12 Loops 6÷0 (left) and 8÷1 (right), implemented using counter 74192

When using 74162 and 74163 counters, only count up loops are possible, in range 0÷9 or 0÷15, respectively. The end of loop is detected, followed by a reset or load with the start of loop. Since reset and load are synchronous, their effect is only after the next rising edge of the clock, thus avoiding the overwrite of the end of loop. The implementation of the loop $D_{3:0}$÷*Val*, using the counter 74163, is presented in Figure 10. 13, along with the actual signal values, when the loop is restarted.

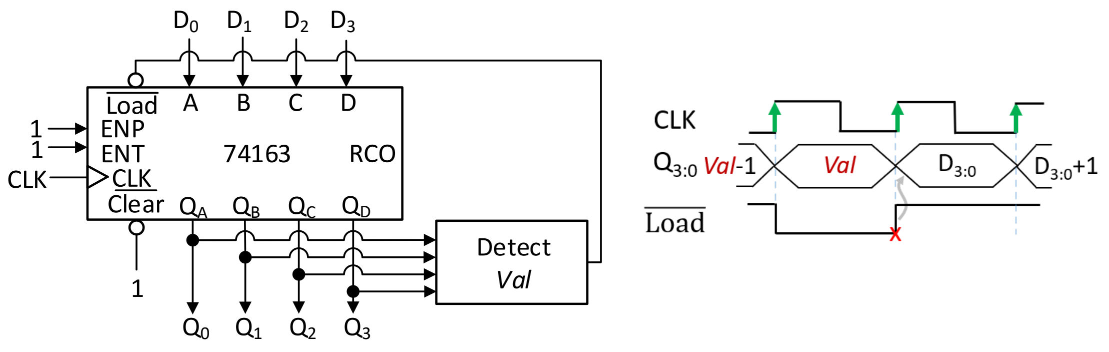


Figure 10. 13 Implementing loop $D_{3:0}$÷*Val* using counter 74163 (left), and the signal values, when the loop is restarted (right)

To implement the loop 10÷12, using counter 74163, the value 10 is loaded when 12 is detected on output. The test circuit is highlighted in Figure 10. 14.

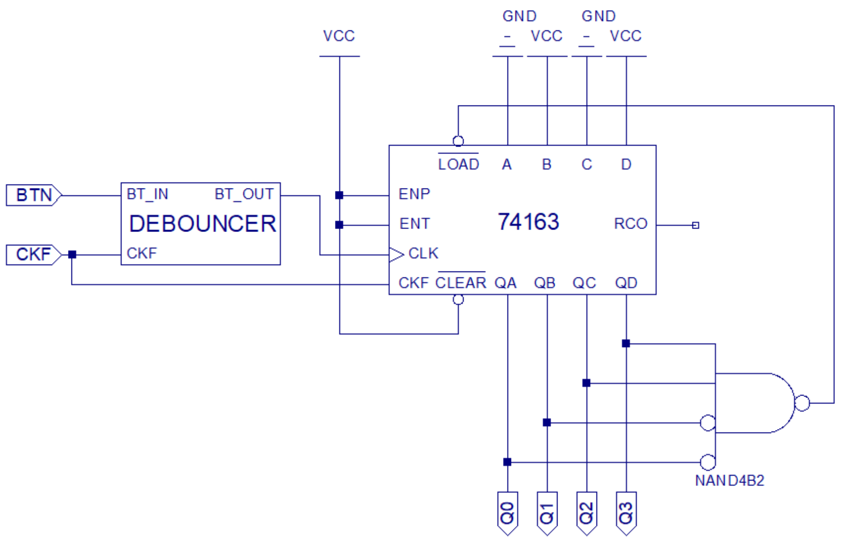


Figure 10. 14 Test circuit for the counter with loop 10÷12, implemented using 74163

The following circuits implement the count up loops 4÷9 and 0÷8, using the decimal counter 74162:

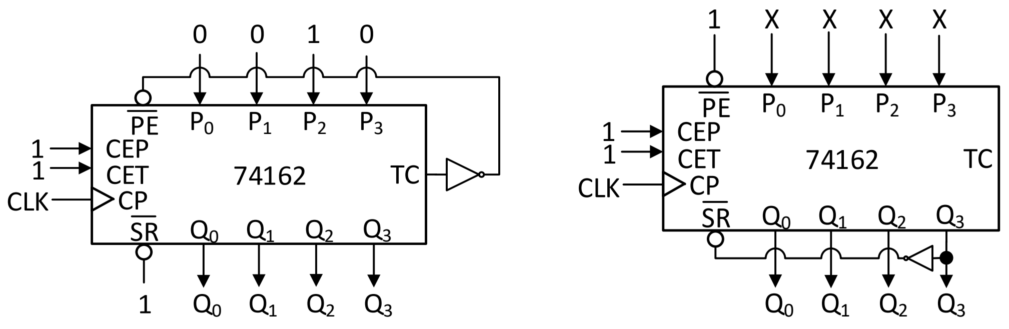


Figure 10. 15 Counters with loops 4÷9 (left) and 0÷8 (right), implemented with 74162

Note: In the first case, the value 9 is detected using the TC flag. In the second case, value 8 is detected using only Q_3, because it is activated solely when reaching 8 within the loop. For the rest of the loop Q_3=0.

When implementing extended loops, the counters can be cascaded onto a larger number of bits. For instance, the loop 37÷74 can be implemented using two cascaded 74193 counters, by loading value 37 (00100101_2) when detecting value 75 (01001011_2):

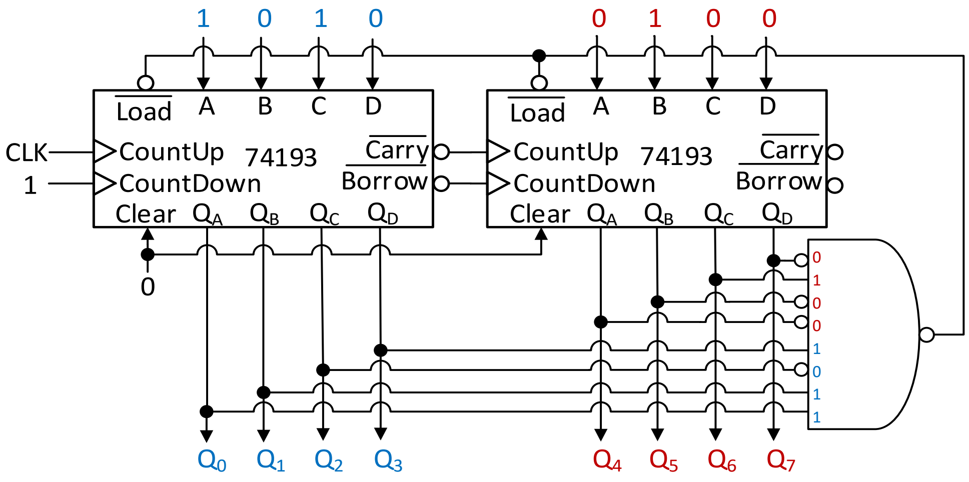
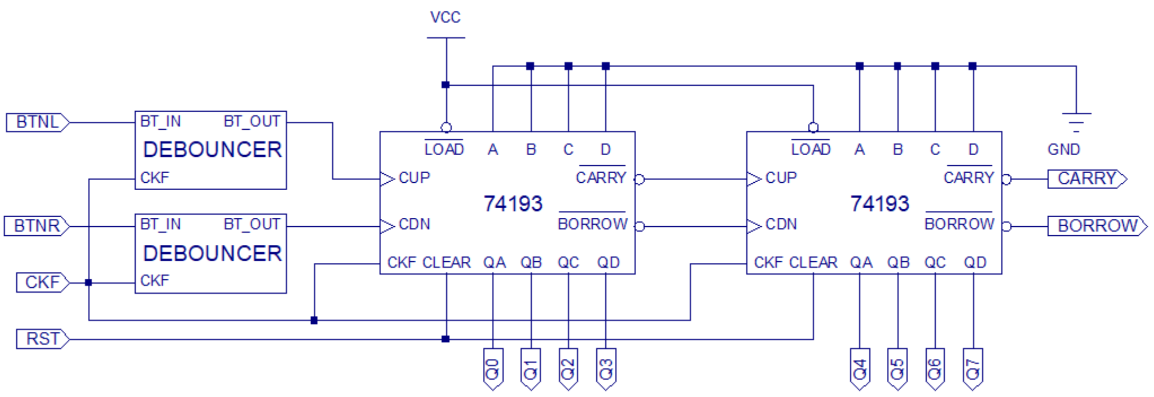


Figure 10. 16 Counter with loop 37÷74, implemented using cascaded 74193 units

10.3 Assignments

1. Implement on the board the counter below, using two 74193 cascaded units. Test the asynchronous reset command, and the up-down counting loop 0÷255.



2. Implement on the board the 4-bit *modulo 3* counter, with ascending counting loop 10÷12, implemented using the 74163 unit. Test the counting loop.
3. Implement in Logisim an 8-bit counter, by cascading 74162 and 74163 units. Test the synchronous reset and load, and the ascending counting loop 0÷159.
4. Implement in Logisim the 4-bit *modulo 7* counter, with descending counting loop 6÷0, implemented using the 74192 unit. Test the counting loop.
5. Implement in Logisim the 4-bit *modulo 6* counter, with ascending counting loop 4÷9, implemented using the 74162 unit. Test the counting loop.
6. Implement in Logisim the 4-bit *modulo 38* counter, with ascending counting loop 37÷74, by cascading two 74193 units. Test the counting loop.