Logic design — 7

Designing synchronous and asynchronous counters

by B. Holdsworth* and D. Zissos†

*Chelsea College, University of London †Dept. of Computing Science, University of Calgary, Canada.

Counters are cyclic sequential circuits which return to their initial state after a specified number of changes in the input state. The output of a counter in its specified code gives the number of changes of the input signal or the number of input pulses received since the circuit was last in its initial state. Counters are being used extensively in industrial plants for such functions as controlling the position of a machine tool or for packing a specified number of items in a box. They are also used in laboratory environments for such functions as counting frequency, recording time, speed and acceleration.

Codes

The most commonly used codes in electronic counters are:

- True binary (8-4-2-1) code,
- Gray codes,
- B.c.d. codes and
- Ordered codes, for example the excess-3 (XS-3).,

The true binary code, often referred to simply as the "binary code" is the simplest because each digit is represented in a conventional binary system. Gray codes are those in which adjacent numbers differ in one bit only, eliminating races which arise when two or more bits attempt to change simultaneously. The true binary code is shown in Table 1, for four binary digits.

If all the sixteen combinations in the sequence in Table 1 are used, the counter is called a maximum-length counter; if, on the other hand, only the first ten combinations are used the counter is called a scale-of-ten counter.

A Gray code in which only one digit changes at a time is called a single-step code, the best known one being the reflected binary code. This code is tabulated in Tables 2(a) and 2(b) for both three and four binary digits. Examination of Table 2(a) shows that reflection of the three least significant digits takes place about the centre line of the code. All those combinations above the centre line have a most significant digit of 0 whilst those below have a most significant digit of 1. Similar comments can be made about the three-digit code except that, in this case, reflection of the two least significant digits takes place.

The sequence of the 4-bit reflected binary code is shown plotted on a Karnaugh map in Fig. 1(a). The plot shows that, as the code proceeds from one combination to the next, only one cell boundary is crossed. It is clear that any single-step Gray code can be developed immediately from a Karnaugh map by tracing a single step path through the map as shown in Fig. 1(b). The code sequence for this example is shown in Fig. 1(c).

d	D	C	B	A
dec.	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Unused code combination in decade counters

Table 1. True binary code, with unused combinations for decade counters.

In b.c.d. (binary-coded-decimal) codes, each of the ten decimal digits 0 to 9, is represented by a binary code, frequently the 8-4-2-1 code. For example the b.c.d. (8-4-2-1) representation of 456 is 0100, 0101, 0110. B.c.d. codes provide a useful link between the counting systems used by digital machines and those used by human beings.

The codes tabulated in Tables 3(a) and 3(b) are examples of weighted b.c.d. codes.

In a weighted code a weight W_j is assigned to the j^{th} binary digit. For example, for the 8-4-2-1 code combination 1001, $W_4 = 8$, $W_3 = 4$, $W_2 = 2$ and $W_1 = 1$

Hence,

$$Z_{dec} = \sum_{j=1}^{j=4} W_j S_j$$

d	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

Reflection

d	C	B	A
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Reflection

Table 2. Four-bit reflected binary (a) and three-bit (B) reflected binary code.

d	D	C	B	A
	2	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

d	D	C	B	A
	5	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

Table 3. Weighted codes. 2-4-2-1 code is at (a) while (b) shows 5-4-2-1 code.

d	D	C	B	A
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

Table 4. Excess-3 code (XS-3).

d	D	C	B	A	p
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1

d	D	C	B	A	p
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0

Table 5. Parity. 8-4-2-1 code at (a) has extra bit to give odd parity and that at (b) has even parity.

where S_j is the value of the j^{th} binary digit, and

$$Z_{dec} = 1\times8+0\times4+0\times2+1\times1=9.$$

The various code combinations in the 2-4-2-1 and the 5-4-2-1 codes can be evaluated in a similar manner.

In an ordered code, the various combinations are assigned to the different decimal digits by means of a mathematical equation. An example of this is the XS-3 code. For this code

$$Z_{dec}=\sum_{j=1}^{j=4} W_j S_j - 3, \quad \text{where}$$

$$W_4=8, W_3=4, W_2=2, W_1=1.$$

Hence, the code combination 0100 = (0 × 8 + 1 × 4 + 0 × 2 + 0 × 1) − 3 = 1. The XS3 code is shown tabulated in Table 4.

Codes can be made error-detecting by the addition of extra bits, called parity bits. In Table 5(a) the 8-4-2-1 code has an additional bit in the column headed *p* which establishes odd parity in each code combination, i.e., each code combination contains an odd number of 1's. Similarly in Table 5(b) a parity bit has been added to the same code which, in this instance, establishes even parity for each code combination. Detection equipment is now required at the receiving end which, in the case of odd parity, is used to determine whether each code combination has an odd number of 1's.

Codes can also be made error-correcting by the addition of extra bits whose function is to detect an error and its position. The most important codes of this kind are the Hamming codes, in which the bit positions are numbered in sequence from left to right. Those positions numbered as a power of 2 are reserved for parity check bits, whilst the remaining positions are used for the information bits.

For a seven bit code combination:

1 2 3 4 5 6 7

p_1 p_2 x_3 p_4 x_5 x_6 x_7

p_1, p_2 and p_4 are the parity bits and x_3, x_5, x_6 and x_7 are the information bits. The parity bits are obtained from the information bits as follows:

p_1 is selected to establish even parity over bits 1,3, 5 and 7

p_2 is selected to establish even parity over bits 2, 3, 6 and 7

p_4 is selected to establish even parity over bits 4, 5, 6 and 7

The Hamming code combinations for the natural n.b.c.d. code are shown below in Table 6.

d	p_1	p_2	x_3	p_4	x_5	x_6	x_7
0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1
2	0	1	0	1	0	1	0
3	1	0	0	0	0	1	1
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1
6	1	1	0	0	1	1	0
7	0	0	0	1	1	1	1
8	1	1	1	0	0	0	0
9	0	0	1	1	0	0	1

Table 6. Hamming combinations for n.b.c.d. code.

The correction process for this code is carried out on the assumption that only one bit is in error and that it is only necessary to locate that bit. This is achieved by checking for odd parity over the same three code combinations for which even parity was established at the transmitting end. The check is carried out with the aid of the exclusive–OR function.

For the exclusive–OR function $A \oplus B = \bar{A}B + A\bar{B}$ and hence

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

The above tabulation shows that the value of the exclusive-OR function is 1 when either A or B are 1, and is 0 when both A and B are either 0 or 1. In other words the value of the exclusive-OR function is 1 when odd parity exists.

The check functions are:

$$c_1 = p_1 \oplus x_3 \oplus x_5 \oplus x_7$$
$$c_2 = p_2 \oplus x_3 \oplus x_6 \oplus x_7$$
$$c_4 = p_4 \oplus x_5 \oplus x_6 \oplus x_7$$

If $c_1 = 1$ there must be an error in p_1, x_3, x_5 or x_7. The bit in error, E, may be obtained from the table below

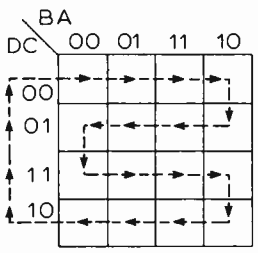
c_4	0	0	0	0	1	1	1	1
c_2	0	0	1	1	0	0	1	1
c_1	0	1	0	1	0	1	0	1
E	0	1	2	3	4	5	6	7

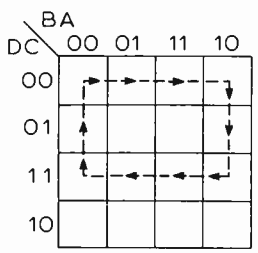
For example, suppose the code combination received is 1101101. Then $c_1 = 1$, $c_2 = 0$ and $c_4 = 1$. Hence the 5th bit is in error and the code combination should read 1101001.

Synchronous counters

The design steps for synchronous counters are (1) draw a state diagram, (2) code the states with the selected counting code, and (3) derive the input equations for the counter flip-flops.

Binary counters (maximum length). For the sake of consistency, variable A is assigned to the 2^0 bit, B to the 2^1 bit, C





d	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	1	1	1	0
6	1	1	1	1
7	1	1	0	1
8	1	1	0	0
9	0	1	0	0

Fig. 1. Karnaugh plots of reflected binary (a) and Gray code (b). Tabulation of Gray code is at (c).

to the 2^2 bit and so on. In deriving the general form of maximum-length binary counters, use will be made of the fact that the addition of higher order counting stages does not affect the lower order counting stages. This, of course, is also the case in conventional decimal counts – for example, the "units" and "tens" of a car odometer change at the end of every one and ten miles travelled, irrespective of the number of stages in the odometer.

Scale-of-2 'up' counter. Figure 2(a) shows the state diagram and codes.
The flip-flop equations are:
$S_A = S_0 = \bar{A}$, therefore, $J_A = 1$
$R_B = S_1 = A$, therefore, $K_A = 1$
The corresponding circuit is shown in Fig. 2(b)

Scale-of-4 'up' counter. $J_A = K_A = 1$, as for a scale-of-2 counter. The state diagram and codes are in Fig. 3(a). The flip-flop equations are:
$S_B = S_1 + (S_2) = AB$, therefore, $J_B = A$
$R_B = S_3 + (S_0) = AB$, therefore, $K_B = A$
The corresponding circuit is shown in Fig. 3(b).

Scale-of-8 'up' counter. $J_A = K_A = 1$ and $J_B = K_B = A$, as for the scale-of-4 counter. The state diagram and codes are in Fig. 4(a) and the flip-flop equations are;
$S_C = S_3 + (S_4) + (S_5 + (S_6) = AB\bar{C}$, therefore, $J_C = AB$
$R_C = S_7 + (S_0) + (S_1) + (S_2) = ABC$, therefore, $K_C = AB$
The corresponding circuit is shown in Fig. 4(b).

Scale-of-2^n 'up' counter. By observation, the flip-flop equations are;
$J_A = K_A = 1$
$J_B = K_B = A$
$J_C = K_C = AB = BJ_B$
$J_D = K_D = ABC = CJ_C$
$J_E = K_E = ABCD = DJ_D$ and so on.
If speed is essential, large input gates must be used to implement directly the functions in the third column.

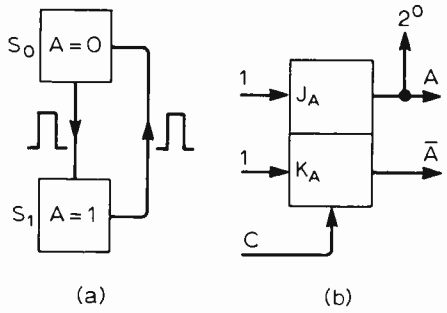
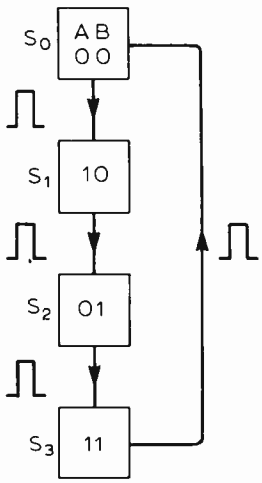


Fig. 2. State diagram for one-stage (scale-of-two) counter (a) and its circuit realization (b).



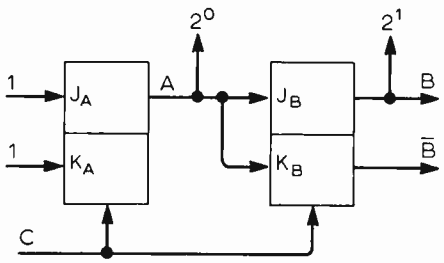


Fig. 3. Two-stage (scale-of-four) counter state diagram and codes (a) and circuit embodiment (b).

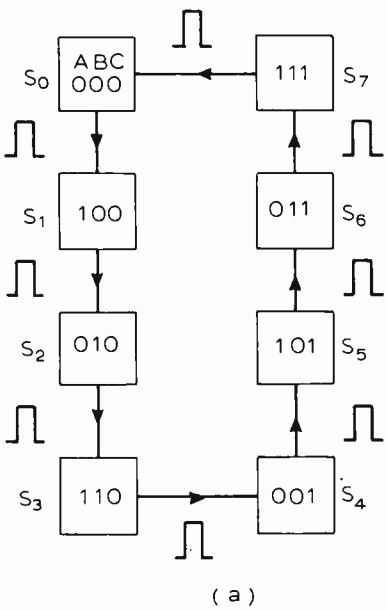
Synchronous 'down' binary counters (maximum length) can be designed in precisely the same manner and the following flip-flop equations are obtained.
$J_A = K_A = 1$
$J_B = K_B = \bar{A}$
$J_C = K_C = \bar{A}\bar{B} = \bar{B}J_B$
$J_D = K_D = \bar{A}\bar{B}\bar{C} = \bar{C}J_C$ and so on

Note that in the case of binary counters it is possible to use an 'up' counter to count down by utilizing the complementary flip-flop outputs as shown in Table 7.

d	C	B	A	d	$\bar{C}$	$\bar{B}$	$\bar{A}$
0	0	0	0	7	1	1	1
1	0	0	1	6	1	1	0
2	0	1	0	5	1	0	1
3	0	1	1	4	1	0	0
4	1	0	0	3	0	1	1
5	1	0	1	2	0	1	0
6	1	1	0	1	0	0	1
7	1	1	1	0	0	0	0

Table 7. Using the complementary outputs of a chain of flip-flops to count down.

The next part of this article will continue the treatment of counters, going on to discuss Gray code types, up-down counters and their control and ripple-through counters.



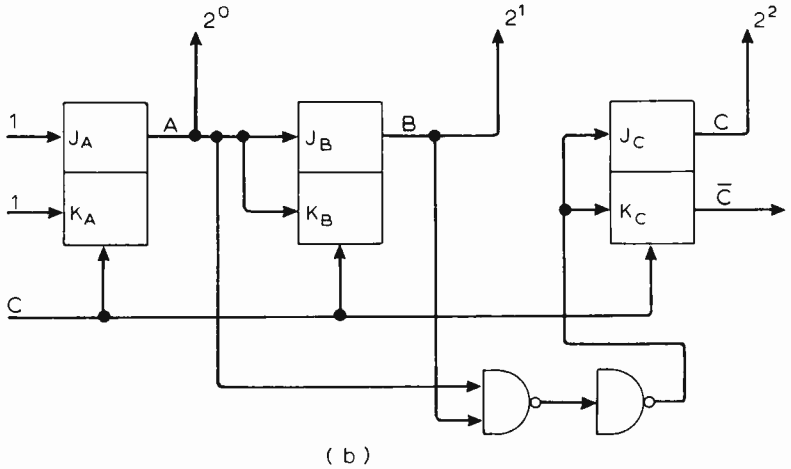


Fig. 4. State diagram (a) and circuit (b) of three-stage (scale-of-eight) counter.