

Voted Glitch-Free Counters

R. Barto

Spacecraft Digital Electronics, El Paso, Texas

Abstract

This paper investigates three counter types and compares their bit-flip immunity and ability to be decoded without glitches. A cascaded Johnson ring counter is presented having both attributes.

I. INTRODUCTION

In an environment where single-event bit flips may be expected, from cosmic rays, for example, a desirable property of counters is that their count values be immune to bit flips. This requires bit redundancy and mechanisms that check for and correct errors. Another desirable counter property is that its states be glitchlessly decodable, allowing the decoded counts to be sourced directly to edge-triggered inputs without having to be “cleaned up” with a following flip-flop which would also have to be protected from bit flips

Of interest in this paper are counters required to have both properties. The properties are orthogonal in that bit-flip immunity is dependent on error checking and correcting mechanisms, while glitch-free decodability is a function of counter topography. Three types of counters are discussed here: a counter based on Hamming codes, a triple-modular redundant (TMR) binary counter, and a TMR Johnson ring counter. The advantages and disadvantages of each counter are discussed, and a cascaded ring counter having both single bit-flip immunity and partial glitch-free decoding is presented.

II. SINGLE BIT-FLIP IMMUNE COUNTERS

A counter that is immune to single bit-flips must incorporate single bit error correction that operates in real time. Any correction mechanism that on detection of an error stops, resets, or returns the counter to a previous count in order to correct the error would not be acceptable.

A. Hamming Code Counters

A counter with single bit error correction can be constructed by incorporating a Hamming code. For any $m \geq 2$ there is a Hamming code with each word having $2^m - 1$ total bits and $2^m - 1 - m$ information bits. Figure 1 shows a Hamming code counter for $m=3$. Each of the check bits check parity on a subset of the information bit. A further discussion of Hamming codes can be found in [1]. The equations governing its operation are as follows.

Next Parity Logic:

$$\begin{aligned} NP0 &= NC2 \oplus NC1 \oplus NC0 \\ NP1 &= NC3 \oplus NC2 \oplus NC1 \\ NP2 &= NC3 \oplus NC1 \oplus NC0 \end{aligned}$$

Next Count Logic:

$$\begin{aligned} NC0 &= \neg Cout0 \\ NC1 &= NC1 \oplus Cout0 \\ NC2 &= NC2 \oplus (Cout1 \bullet Cout0) \\ NC3 &= NC3 \oplus (Cout2 \bullet Cout1 \bullet Cout0) \end{aligned}$$

Correction Logic:

$$\begin{aligned} S0 &= C2 \oplus C1 \oplus C0 \oplus P0 \\ S1 &= C3 \oplus C2 \oplus C1 \oplus P1 \\ S2 &= C3 \oplus C1 \oplus C0 \oplus P2 \end{aligned}$$

$$\begin{aligned} E0 &= S0 \bullet (\neg S1) \bullet S2 \\ E1 &= S0 \bullet S1 \bullet S2 \\ E2 &= S0 \bullet S1 \bullet (\neg S2) \\ E3 &= (\neg S0) \bullet S1 \bullet S2 \end{aligned}$$

$$\begin{aligned} Cout0 &= C0 \oplus E0 \\ Cout1 &= C1 \oplus E1 \\ Cout2 &= C2 \oplus E2 \\ Cout3 &= C3 \oplus E3 \end{aligned}$$

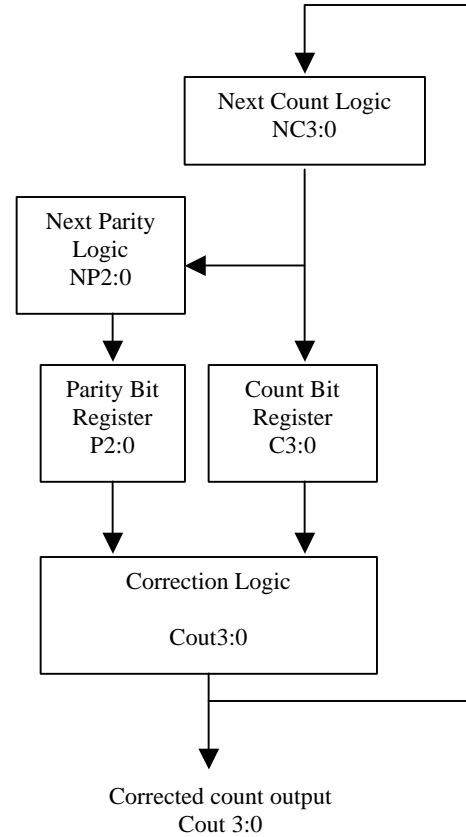


Figure 1: Hamming Code Counter

Table 1 gives the number of parity bits and information bits for various values of m . The ranges given for information

bits arise because it is not necessary to use all the information bits possible for a given value of m.

Table 1: Hamming Code Sizes

m, Check Bits	Total Bits	Information Bits
2	3	1
3	7	2 to 4
4	15	5 to 11
5	31	12 to 26
6	63	27 to 57
7	127	58 to 120

B. Triple-Modular Redundant (TMR) Counters

The Hamming code size of m=2 has the same bit count as TMR, in which each counter bit is represented by 3 flip-flops whose outputs are majority voted as shown in Figure 2.

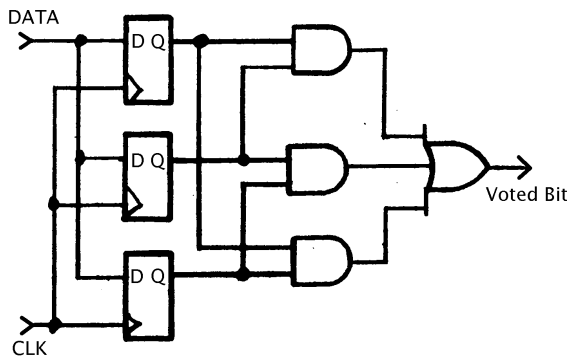


Figure 2: TMR Bit Structure

The 4-bit counter of figure 1 can be implemented using TMR as shown in Figure 3. The TMR register contains 4 copies of

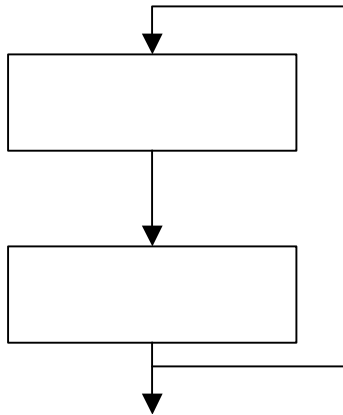


Figure 3: 4-Bit TMR Binary Counter

the Figure 2 structure, and the Next Count Logic is the same as in Figure 1. To compare the relative size of the two counters, the gate counts of Figures 1 and 3 are given in

Tables 2 and 3 respectively. For a fair comparison, all logic is assumed to be implemented in 2-input gates and D flip-flops.

Table 2: 4-bit Hamming Counter Gate Count

Logic	AND	OR	XOR	INV
NP2:0			2*3=6	
NC0				1
NC1			1	
NC2	1		1	
NC3	2		1	
S2:0			3*3=9	
E0	2			1
E1	2			
E2	2			1
E3	2			1
Count3:0			4	
totals	11	0	22	4
Total Combinational Gates				37
Total Flip-flops				7
Total Device Count				44

Table 3: 4-bit TMR Counter Gate Count

Logic	AND	OR	XOR	INV
NC0				1
NC1			1	
NC2	1		1	
NC3	2		1	
Count3:0	3*4=12	4*2=8		
totals	15	8	3	1
Total Combinational Gates				27
Total Flip-flops				12
Total Device Count				39

Note that neither the TMR nor Hamming counters shown here include a mechanism for correcting a flipped bit, other than the updating that occurs during normal counter operation. The update rate for all bits is assumed sufficient to make the probability of double-bit errors acceptably small.

C. Comparing Hamming and TMR Counters

The gate counts of the two counters are comparable, with the Hamming counter having the advantage of lighter clock loading. For a 16-bit counter the gate count difference is smaller, but the clock loading for the Hamming counter is less than half the loading of the TMR counter, as shown in Tables 4 and 5.

Another difference is the distance in gate levels between the counter flip-flops and the counter outputs. Regardless of the size of the counter, a TMR counter's outputs are 4 gate levels from the flip-flop outputs (again considering 2-input gates; this can obviously be optimized to 3 gate levels, depending on the technology). In the 4-bit Hamming counter, this (unoptimized) distance is 5 gate levels when the longest path through the correction logic is considered (which includes more complex XORs), and 8 levels in the 16-bit counter. Since the Next Count Logic is the same for TMR and Hamming counters, the TMR counter would be the faster of the two, if its extra clock loading has no effect.

Table 4: 16-bit Hamming Counter Gate Count

Logic	AND	OR	XOR	INV
NP4:0			35	
NC15:0	14		15	1
S4:0			40	
E15:0	16			3
Cout15:0			16	
Totals	30	0	106	4
Total Combinational Gates				140
Total Flip-flops				21
Total Device Count				161

Table 5: 16-bit TMR Counter Gate Count

Logic	AND	OR	XOR	INV
NC15:0	14		15	1
Cout15:0	48	32		
totals	62	32	15	1
Total Combinational Gates				110
Total Flip-flops				48
Total Device Count				158

A further difference between Hamming and TMR counters is that TMR counters are more strongly single bit-flip immune. In a Hamming counter, a flip in the Count Bit register takes two paths to the corrected bit output in the correction logic. Referring to the equations for Figure 1, each C_i goes directly to the XOR gates producing C_{outi} , and through the network producing the E_i , which would be the longer path for any implementation. A flipped bit will thus create an incorrect C_{out} for a short time before it is corrected, allowing the possibility of an incorrect next count being loaded into the register, depending on the timing of the flip. In a TMR counter this could not happen since one bit flipping has no effect on the voter output.

III. GLITCH-FREE DECODABLE COUNTERS

Decoding states of counters which involve functions of bits undergoing opposite transitions creates the possibility of decoder glitches. For example, the ANDing of two bits that at some point will transition from 01 to 10 may produce an undesirable glitch if the intermittent state 11 occurs. For this reason, glitch-free decoding can only be guaranteed if only one bit at a time is allowed to change after any clock edge. If two bits change from, for example, 11 to 00, the intermediate states of 10 or 01 might occur and be decoded erroneously. The most common clock structure operating in the one-bit-at-a-time manner is the Johnson ring counter. The count sequence for a 4-bit ring counter is given in Table 6, and a schematic of the counter is given in Figure 4.

The count sequence allows any count to be decoded with only one AND gate (assuming the Q^* flip-flop outputs are available), and since only one bit changes on any count transition, the decoding gate cannot glitch. The disadvantage of the sequence is that ring counters exhibit poor flip-flop usage. Table 7 compares the number of states available with ring counters ($2n$) and binary counters (2^n).

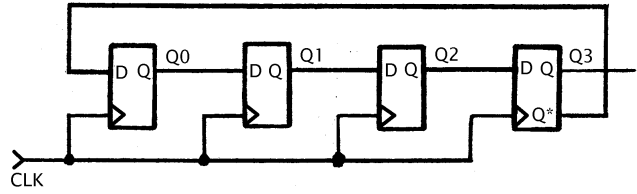


Figure 4: 4-Bit Johnson Ring Counter

Table 6: 4-bit Ring Counter Sequence

Count	Counter Bit Values $Q0 \cdot Q1 \cdot Q2 \cdot Q3$
0	0000
1	1000
2	1100
3	1110
4	1111
5	0111
6	0011
7	0001

Table 7: Comparison of Ring and Binary Counters

Bit Count	Ring Counter	Binary Counter
2	4	4
4	8	16
8	16	256
16	32	65536

IV. GLITCH-FREE, SINGLE BIT-FLIP IMMUNE COUNTERS

A TMR ring counter can be made by replacing each of the flip-flops in Figure 4 with the TMR flip-flop of Figure 2. An inversion is required at the last bit output for input to the first bit. The TMR counter would require 12 flip-flops, 12 AND gates, 6 OR gates, and one inverter or NOR gate, assuming 2-input gates.

Referring to Table 7, a bit count of 2 provides 4 states for either a binary or a ring counter. Using the 2-bit counter structure shown in Figure 5, a cascaded ring counter can be constructed for any number of bits, and it would have the same bit usage as a binary counter. A 16 bit counter would require 8 such stages (giving $4^8=2^{16}$ count states) and would require the device count shown in Table 8, which assumes that each flip-flop is a TMR set as shown in Figure 2. Assuming a free running counter, the first stage requires only the 2 flip-flops and one C_{out} AND gate. The last stage would not have the 2 C_{out} AND gates.

The increased device count for this counter might be offset by the decreased gate count required for decoding count states. Unfortunately, the counter as a whole does not retain glitch free decodability when several stages must be included in a decoding circuit, since several bits can be expected to change after each clock edge. Each stage, however, can be glitchlessly decoded.

