9-Bit parity generator/checker with PLS153/153A AN021

INTRODUCTION

This application note presents the design of a parity generator using Philips Semiconductors PLD, PLS153 or PLS153A, which enables the designers to customize their circuits in the form of “sum- of-products”. The PLA architecture and the 10 bi-directional I/O’s make it possible to implement the 9-bit parity generator/checker in one chip without any external wiring between pins.

The parity of an 8-bit word is generated by counting the number of “1’s” in the word. If the number is odd, the word has odd parity. If the number is even, the word has even parity. Thus, a parity generator designed for even parity, for example, will generate a “0” if the parity is even, or a “1” if parity is odd. Conversely, an odd parity generator will generate a “0” if the parity of the word is odd, or a “1” if the parity is even. This bit is then concatinated to the word making it 9-bits long. When the word is used elsewhere, its parity may be checked for correctness.

FEATURES

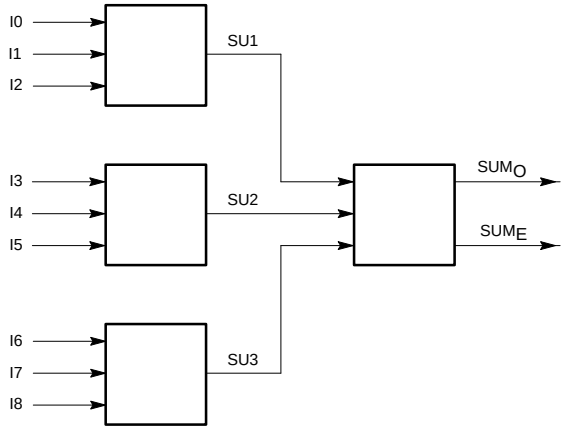
- Generates even and odd parities (SUME and SUMO)
- SUME = “1” for even parity, “0” for odd parity
- SUMO = “1” for odd parity, “0” for even parity
- Generate parity or check for parity errors
- Cascaded to expand word length

DESCRIPTION

The most straightforward way of implementing the parity generator/checker is to take the 9-input truth table (8 inputs for the 8-bit word, and 1 input for cascading the previous stage) and put it in a 256×4 PROM. Since there are 2^9 combinations and half of them is odd, the other half is even, the circuit will take 256 terms.

An alternative is to divide the 9-bits into 3 groups of 3-bits as shown in Figure 1. If the sum of the 3-bits is odd, then the intermediate output SU1, or SU2, or SU3 equals 1. Otherwise it equals 0. The intermediate results are grouped together and SUMO becomes “1” if the sum is odd, otherwise SUMO equals “0”.

The circuit is implemented using SNAP as shown in Figure 3. SU1 is an intermediate output for inputs I_0, I_1 and I_2. In the same manner, SU2 and SU3 are intermediate outputs for I_3, I_4, I_5 and I_6, I_7, I_8.



I_0	I_1	I_2	SU1
0	0	0	0
0	0	1	1
0	1	1	0
0	1	0	1
1	0	1	0
1	0	0	1
1	1	0	0
1	1	1	1

I_3	I_4	I_5	SU2
0	0	0	0
0	1	1	0
0	0	1	1
0	1	0	1
1	0	1	0
1	1	0	0
1	0	0	1
1	1	1	1

I_6	I_7	I_8	SU3
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	0
0	0	1	1
0	1	0	1
1	0	0	1
1	1	1	1

SU1	SU2	SU3	SUMO	SUME
0	0	0	0	1
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	0

Figure 1. Block Diagram of 9-Bit Parity Generator/Checker

RESOURCES

The design uses up 20 product terms and 5 control terms leaving 12 product terms and 4 bi-directional I/O’s to implement other logic designs.

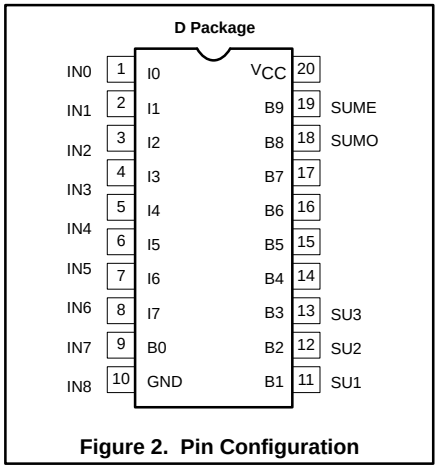


Figure 2. Pin Configuration

9-Bit parity generator/checker with PLS153/153A AN021

```
@PINLIST                This circuit is a 9-bit parity generator/checker commonly used for error detection in high speed data
                        transmission/retrieval.
  in[0..8]      i;      The odd parity output (SUMO) is high when the sum of the data bits is odd.  Otherwise it is low.
  su[1..3]      o;      The even parity output (SUME) is high when the sum of the data bits is even.  Otherwise it is low.
  sumo          o;
  sume          o;      SU3, SU2 and SU1 are intermediate terms.
                        This design was done using the Truth Table Entry method of Philips Semiconductors SNAP
                        software.
@TRUTHTABLE
    [IN2,   IN1,    IN0  :  SU1]
     0      0       0    :  0 ;
     0      0       1    :  1 ;
     0      1       0    :  1 ;
     0      1       1    :  0 ;
     1      0       0    :  1 ;
     1      0       1    :  0 ;
     1      1       0    :  0 ;
     1      1       1    :  1 ;

    [IN5,   IN4,    IN3  :  SU2]
     0      0       0    :  0 ;
     0      0       1    :  1 ;
     0      1       0    :  1 ;
     0      1       1    :  0 ;
     1      0       0    :  1 ;
     1      0       1    :  0 ;
     1      1       0    :  0 ;
     1      1       1    :  1 ;

    [IN8,   IN7,    IN6  :  SU3]
     0      0       0    :  0 ;
     0      0       1    :  1 ;
     0      1       0    :  1 ;
     0      1       1    :  0 ;
     1      0       0    :  1 ;
     1      0       1    :  0 ;
     1      1       0    :  0 ;
     1      1       1    :  1 ;

    [SU3,   SU2,    SU1  :  SUMO,   SUME]
     0      0       0    :  0       1 ;
     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       0 ;
```

Figure 3. SNAP Truth Table Entry

```
@logic equations
su1     =   ( (in2 * in1 * in0)
        +   (in2 * in1 * in0)
        +   (/in2 * in1 * /in0)
        +   (/in2 * /in1 * in0);

su2     =   ( (in5 * in4 * in3)
        +   (in5 * in4 * in3)
        +   (/in5 * in4 * /in3)
        +   (/in5 * /in4 * in3);

su3     =   ( (in8 * in7 * in6)
        +   (in8 * in7 * in6)
        +   (/in8 * in7 * /in6)
        +   (/in8 * /in7 * in6);

sum0    =   ( (su3 * su2 * su1)
        +   (su3 * /su2 * /su1)
        +   (/su3 * su2 * /su1)
        +   (/su3 * /su2 * su1);

sume    =   ( (su3 * su2 * su1)
        +   (su3 * /su2 * /su1)
        +   (/su3 * su2 * /su1)
        +   (/su3 * /su2 * su1);
```

Figure 4. Expanded Equations