High speed 8-bit parallel to serial converter AN045

INTRODUCTION

A common function in many systems is to convert parallel data into a serial data stream. A microcontroller may be programmed to shift a byte in a register out to a port, but this is a relatively slow procedure. A simple pre-loadable shift register could perform the basic conversion. However, for the function to be complete, additional circuitry to perform handshaking or control of the process is required. The entire function can be made to fit into a low cost Programmable Logic Device (PLD), including control circuitry tailored to meet specific application requirements.

DESCRIPTION

Figure 1 shows the desired waveforms for a typical implementation. First, a reset signal initializes the system and this circuit. Next, the parallel data to be serialized is applied to the device, possibly from a parallel port of a microcontroller, and a write strobe (WRS) signal pulsed. The PLD then raises a flag (BUSY) and puts the data, one bit at a time, on an output (SDAT) under control of a clock signal (CLK). Another output, (SCLK) is an inverted copy of the transmitting clock, ANDed with a control signal, so it only is active when data is actually being sent. It can be used by the receiving device to clock in the serial data.

How does one get a PLD to perform such a function? Preferably this design should fit into a simple, low cost device such as a 22V10 type PLD. A 22V10 has ten outputs which may be individually configured to be registered or combinatorial. It is possible to make a two input multiplexer circuit in front of eight of the D-type flip-flops. It could then be configured to shift data or load parallel data upon a control signal and clock. However, to provide the output control signal BUSY and gate SCLK, a 3-bit counter will be required to indicate when the last bit of data is shifted out. That would bring the total registers in the design up to eleven, one more than a 22V10 provides. Additionally, the write strobe (WRS) is a short duration asynchronous signal, so more circuitry is still required to synchronize it with the transmitting clock (CLK).

Another method of serializing data is to use a multiplexer (8 to1 for this example) and a counter. The counter controls which bit is to be output from the multiplexer. A count of zero connects input ID0 to the output, a count of one connects ID1, and so on. This will work only if the parallel input data is held stable throughout the serialization process. For this example, the data is applied from one port of a microcontroller and held stable until after the BUSY signal transitions from high-to-low, so a multiplexer will work for this case. An 8-to-1 multiplexer will use only one output, while the three-bit counter will use three outputs of a 22V10, which leaves us with six outputs for other functions. Let's use this technique to implement this example. Additional outputs are required for signals BUSY, SCLK, and some currently unspecified control signals.

A counter may be constructed very easily using a SNAP syntax equation of: "COUNT.D=COUNT#1H;". The "#" (pound) symbol means addition, the ".D" signifies an input to a D-type of flip-flop, and the "1H" is 1 hexadecimal. So the equation is simply COUNT equals COUNT plus 1. The actual equation in Figure 3 contains another term, but more on that later. In addition to the D inputs of the flip-flops, it is necessary to describe the flip-flops clock and reset connections. Those are listed in lines 57and 58 of Figure 3.

A multiplexer is also very easy to describe using SNAP syntax Boolean equations. For an 8-to-1 multiplexer with output SDAT and inputs ID7-ID0 it is:

```
SDAT = ID0 * (COUNT==0H)
     + ID1 * (COUNT==1H)
     + ID2 * (COUNT==2H)
     + ID3 * (COUNT==3H)
     + ID4 * (COUNT==4H)
     + ID5 * (COUNT==5H)
     + ID6 * (COUNT==6H)
     + ID7 * (COUNT==7H);
```

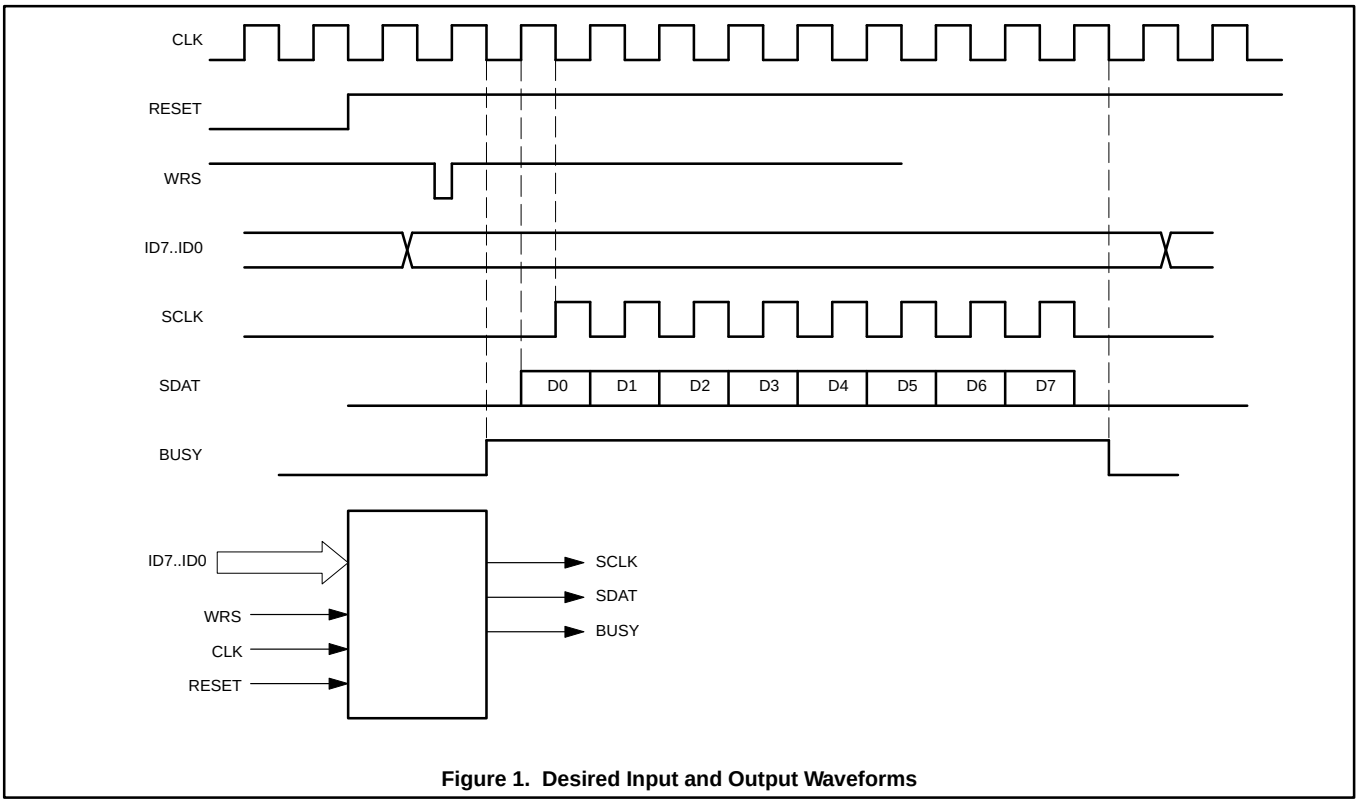

Figure 1. Desired Input and Output Waveforms

High speed 8-bit parallel to serial converter AN045

So far, we have a counter and multiplexer to serialize the data. The process of serialization begins with an asynchronous pulse on the write strobe input (WRS). It is therefore necessary to construct a latch to capture the pulse and then use two registers to synchronize the signal to the input clock. Figure 2 shows the desired operation of two intermediate signals Z and Z1. An extremely simple latch can be made with the equation: "Z = /WRS + Z". Once set with WRS low, it could never be reset. An additional signal named GATE, will be used as an extra term in the latch to reset it. From the waveforms of Figure 2, a table of the three signals may be constructed.

WRS	GATE	Z	Z+
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

	WRS,GATE 00	01	11	10
Z 0	1	1	0	0
1	1	1	0	1

Z+ is the "next state" or what value output Z should be, given the current inputs and the current state of Z. From the table, a Karnough map may be constructed (shown above) and the equation "Z=/WRS+Z*/GATE;" derived.

We have signal Z, which latches the input strobe, but we need to synchronize it to the input clock. That can be done with flip-flop Z1 and the following flip-flop, GATE, described later. For Z1, the equation is simply: "Z1.D=Z;" Z1 is clocked by the rising edge of CLK. Z1's output will go high upon the rising edge of CLK and Z high. It will go low upon a rising edge of CLK and Z low.

According to the original waveforms of Figure 1, a signal named BUSY is required to occur after the falling edge of CLK following a detected strobe (WRS). The internal D-type flip-flops of a 22V10 can only be clocked on the rising edge of the clock, so one of the 22V10's internal flip-flops cannot be used. However a Boolean equation may be used to describe this signal. The times and conditions when this signal is to be high will be noted from Figure 2 and a Boolean expression generated. From Figure 2, at time T2, BUSY should go high. Therefore, one term of the equation for BUSY will be: "Z*Z1*/CLK". When both Z and Z1 are high and CLK is low, then BUSY will go high. This product term will keep BUSY high until time T3.

At time T3, BUSY should remain high and adding a product term of "BUSY*Z1" can keep it high until time T5. This product term actually becomes active long before time T3 arrives, so there will be no glitching of the output. Adding yet another product term of "BUSY*GATE" will keep BUSY high from just after time T3, through time T5, until time T19. Finally, one last product term of "BUSY*CLK", keeps it high until the falling edge of the clock. The combined equation for BUSY is shown in Figure 3 lines 38 through 41.

The last signal to be described is GATE. It is used to control the gating of the inverted clock output SCLK, and also control the already described signals BUSY and Z. GATE can use one of the flip-flops inside the 22V10, as it should only switch after the rising edge of the input clock. It goes high after the first rising edge of CLK after BUSY goes high. Therefore, one of inputs to the GATE flip-flop has to be BUSY. GATE should go low after COUNT reaches seven, so the equation can be "GATE.D = BUSY * /(COUNT==7H);" The input to the GATE flip-flop will be high when BUSY goes high and COUNT is not equal to 7.

Signal GATE was also added to each of the terms in the multiplexer equation and it was added as a term in the counter equation. It was added to the multiplexer so SDAT would be low unless actual data was being sent . It was added to the counter so the counter would only count when GATE was high. This design used nine of the ten possible 22V10 outputs.The input RESET was added to many of the equations to force a proper initialization of the signals. From here it is just a matter of typing the equations into SNAP, running a simulation, and generating a JEDEC file for downloading to a device programmer. Figure 4 shows the SNAP simulation results and Figure 5 shows the associated simulation control language (SCL) file.

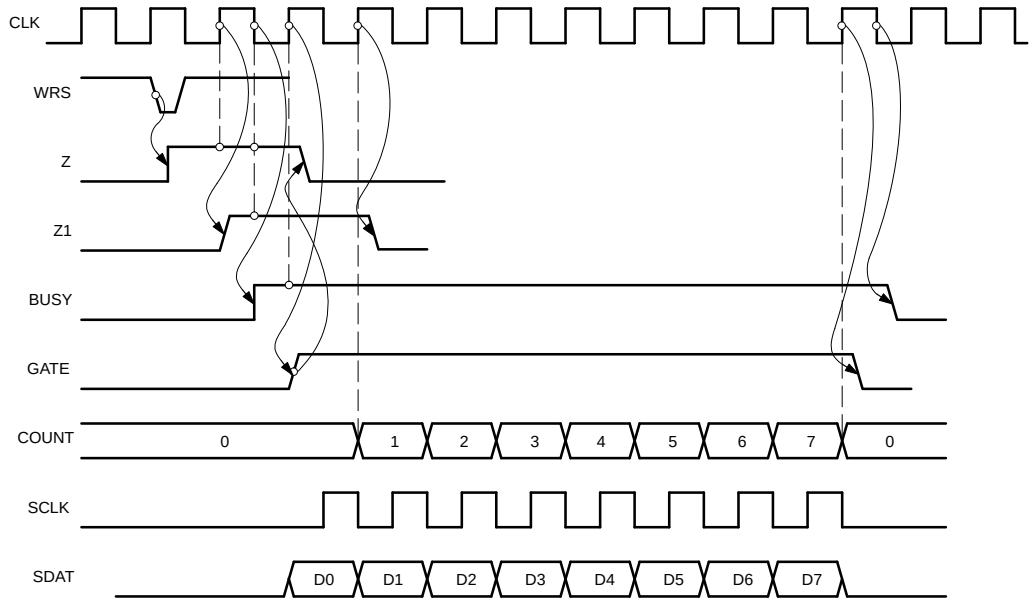

Figure 2. Waveform Timing Relationships

```
 1 |
 2 |"-------------------------------------------------"
 3 |"  High Speed 8-bit Parallel to Serial Converter  "
 4 |"-------------------------------------------------"
 5 |
 6 |@PINLIST
 7 |CLK             I;
 8 |ID[0..7]        I;
 9 |RESET           I;  "active low"
10 |WRS             I;
11 |
12 |BUSY            O;
13 |SCLK            O;
14 |SDAT            O;
15 |C[0..2]         O;
16 |GATE            O;
17 |Z               O;
18 |Z1              O;
19 |
20 |@GROUPS
21 |COUNT=[C2,C1,C0];
22 |
23 |@TRUTHTABLE
24 |@LOGIC EQUATIONS
25 |
26 |    "write strobe latch"
27 |
28 |Z = /WRS*reset+Z*/GATE*reset;
29 |
30 |    "first flip-flop to sychronize WRS to CLK"
31 |
32 |Z1.D   = Z;
33 |Z1.CLK = CLK;
34 |Z1.RST = reset;
35 |
36 |    "busy flag"
37 |
38 |BUSY = Z*Z1*/CLK*reset
39 |     + BUSY*Z1*reset
40 |     + BUSY*GATE*reset
41 |     + BUSY*CLK*reset;
42 |
43 |    "gate for control and 2nd synchronizing flip-flop"
44 |
45 |GATE.D   = BUSY*/(COUNT==7H);
46 |GATE.CLK = CLK;
47 |GATE.RST = reset;
48 |
49 |
50 |     "output clock"
51 |
52 |SCLK = /CLK*GATE;
53 |
54 |     "3-bit up counter"
55 |
56 |COUNT.D   = GATE==1 * COUNT#1H;   "count only when GATE is high"
57 |COUNT.CLK = CLK;
58 |COUNT.RST = reset;
59 |
60 |    "Multiplexer Equations"
61 |
62 |SDAT = ID0*(COUNT==0H)*GATE*reset  "if GATE is low then output a low"
63 |     + ID1*(COUNT==1H)*GATE*reset
64 |     + ID2*(COUNT==2H)*GATE*reset
65 |     + ID3*(COUNT==3H)*GATE*reset
66 |     + ID4*(COUNT==4H)*GATE*reset
67 |     + ID5*(COUNT==5H)*GATE*reset
68 |     + ID6*(COUNT==6H)*GATE*reset
69 |     + ID7*(COUNT==7H)*GATE*reset;
70 |
71 |@INPUT VECTORS
72 |@OUTPUT VECTORS
73 |@STATE VECTORS
74 |@TRANSITIONS
75 |
```

```
                    _______   _______
                   |       \_/       |
                   |                 |
      CLK [ 1]|CLK/I0          VCC|24]
    RESET [ 2]|I1              I/O9|23]
      ID0 [ 3]|I2              I/O8|22] Z1
      ID1 [ 4]|I3              I/O7|21] Z
      ID2 [ 5]|I4              I/O6|20] SDAT
      ID3 [ 6]|I5              I/O5|19] SCLK
      ID4 [ 7]|I6              I/O4|18] GATE
      ID5 [ 8]|I7              I/O3|17] C2
      ID6 [ 9]|I8              I/O2|16] C1
      ID7 [10]|I9              I/O1|15] C0
      WRS [11]|I10             I/O0|14] BUSY
          [12]|GND              I11|13]
              |                 |
              |_________________|
```

NOTE: Line numbers are for reference only, they are NOT part of the design file.

Figure 3. SNAP Listing and Pin File

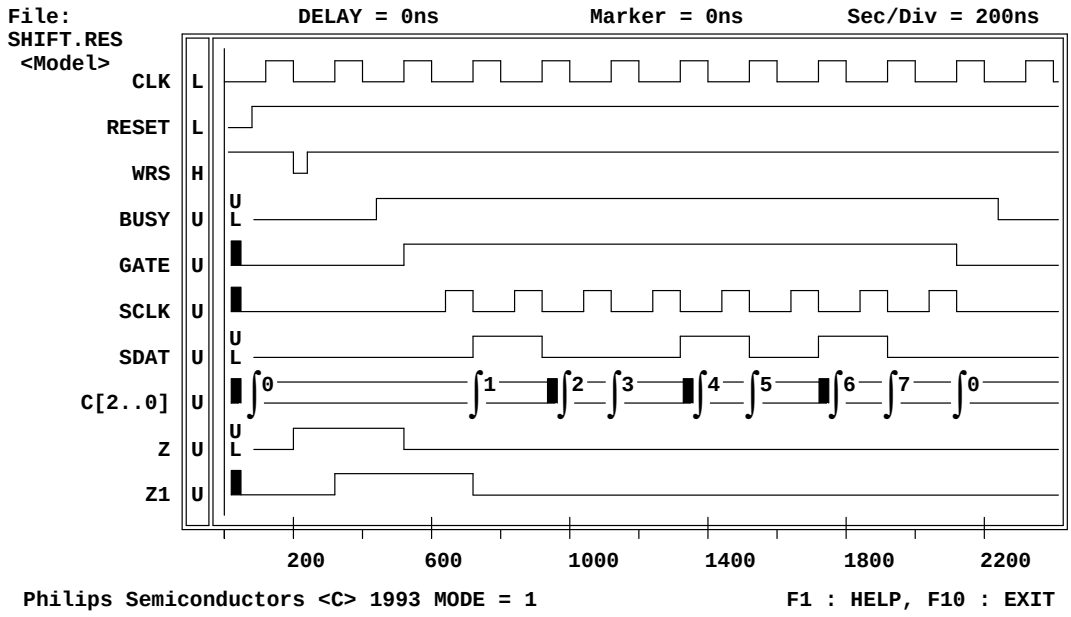

Figure 4. SNAP Simulation Waveforms

```
************************************************
*     Output of Waveform Version 1.90          *
* Date: 04/21/93                 Time: 16:14:48 *
************************************************
*                                              *
* Input File Name  : SHIFT.SCL                 *
* Rule File Name   : Scl Rule                  *
* Output File Name : SHIFT.SCL                 *
*                                              *
************************************************
 P ID0, ID1, ID2, ID3, ID4, ID5, ID6, ID7, CLK, RESET, WRS, BUSY,
#   GATE, SCLK, SDAT, C[2..0], Z, Z1
 PCO
 S 0 (5890) ID0
 S 1 (5890) ID1
 S 0 (5890) ID2
 S 0 (5890) ID3
 S 1 (5890) ID4
 S 0 (5890) ID5
 S 1 (9200) ID6
 S 0 (14290) ID7
 S 0 (100, 200, ETC) CLK
 S 0 (80, 5600, 6000) RESET
 S 1 (180, 240, 4000, 4200) WRS
 SU time = 14290
 F
```

Figure 5. SNAP Simulation SCL File