Innovative Designs with the CY7B991/2/10/20 (RoboClock) Programmable Skew Clock Buffer

Overview

This article discusses various applications of the Cypress Phase Locked Loop-based, skew-defeating clock buffers known as RoboClock. It is assumed that the reader has a working understanding of RoboClock. If not, "Related Articles" shown below are recommended. Unlike traditional clock buffers, RoboClock enjoys the advantages of an internal, multi-tapped PLL, which offers designers two principal advantages: zero propagation delay and configurable phase control, relative to the reference clock.

Zero propagation delay is achieved through the presence of the internal PLL. Because of the properties of PLLs, RoboClock is able to synchronize itself to an incoming reference clock, allowing the buffered outputs to be coincident with the reference input, effectively acting as a zero propagation delay clock buffer. Conventional clock buffer solutions, even those that offer low skew, still have a finite amount of propagation delay. An application example later in this article demonstrates the constraints these delays force designers to operate under, and how RoboClock allows the designer to overcome these constraints.

Configurable phase control of distributed clocks allows the designer to overcome the debilitating effects of clock skew. With today's increasing clock rates, the amount of time required for a clock signal to travel across a circuit board becomes a significant portion of the clock period. This clock skew can result in timing specification violations. RoboClock offers designers the ability to manipulate the phase of the distributed clocks, and thereby compensate for clock skew.

Related Documents

For a more complete description of RoboClock as well as its internal PLL, the reader is encouraged to consult the following documents for additional information:

"CY7B9910/CY7B9920 Low Skew Clock Buffer" data sheet.

"Innovative RoboClock Application" published in the Cypress Semiconductor *Applications Handbook*.

"CY7B991/CYB992 (RoboClock) Test Mode" published in the Cypress Semiconductor *Applications Handbook*.

"Everything You Need to Know About CY7B991 and CY7B992 (RoboClock) But Were Afraid to Ask" published in the Cypress Semiconductor *Applications Handbook*.

"CY7B991/CY7B992 Programmable Skew Clock Buffer" data sheet, published in the Cypress Semiconductor *Data Book*.

Using RoboClock to Overcome a Timing Violation

This design example typifies how RoboClock can be used to solve timing margin problems. In this case, the problem is a register set-up time violation. Represented is an actual design implemented by a major telecommunications manufacturer.

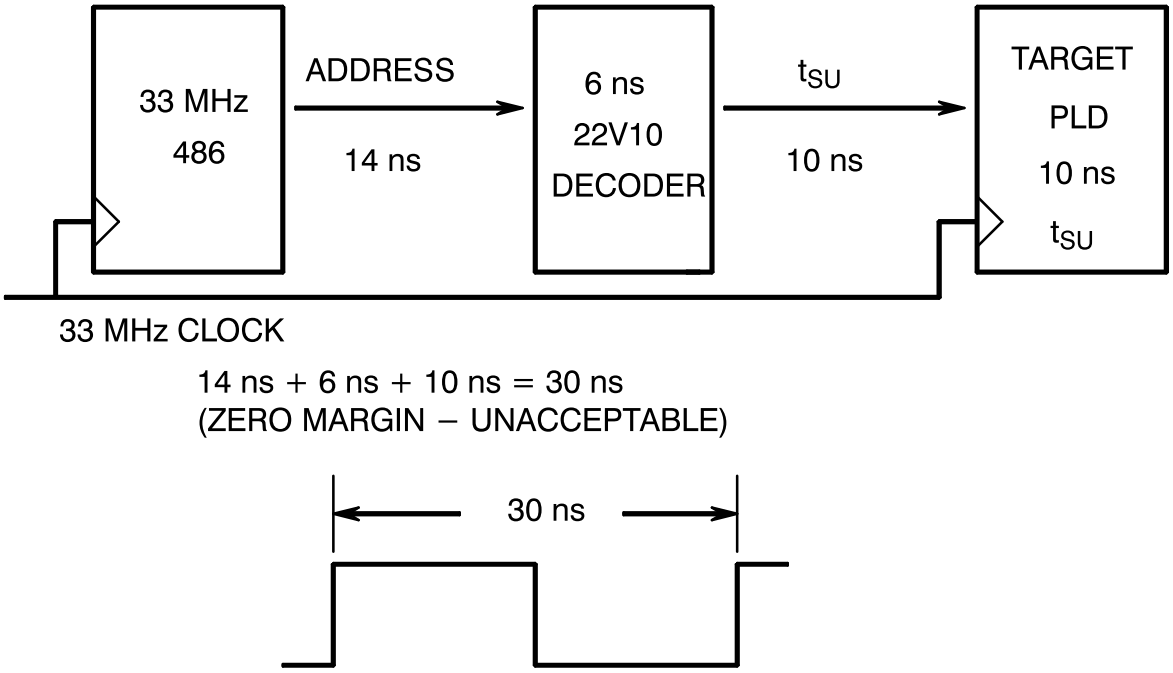

Figure 1. Timing Violation

In this application a 33-MHz 486 microprocessor's address has a critical path to the registered target PLD through a 6-ns 22V10. As shown in *Figure 1*, the address is guaranteed valid out of the 486 processor 14 ns after the initial rising clock edge, after which the address is decoded by a 22V10 (requiring an additional 6 ns) before needing to meet a required 10-ns register set-up time of the CPLD. Examination of the timing constraints shows that no margin is present—a situation deemed unacceptable by the designer.

The designer chose the RoboClock implementation shown in *Figure 2* in order to solve this timing margin

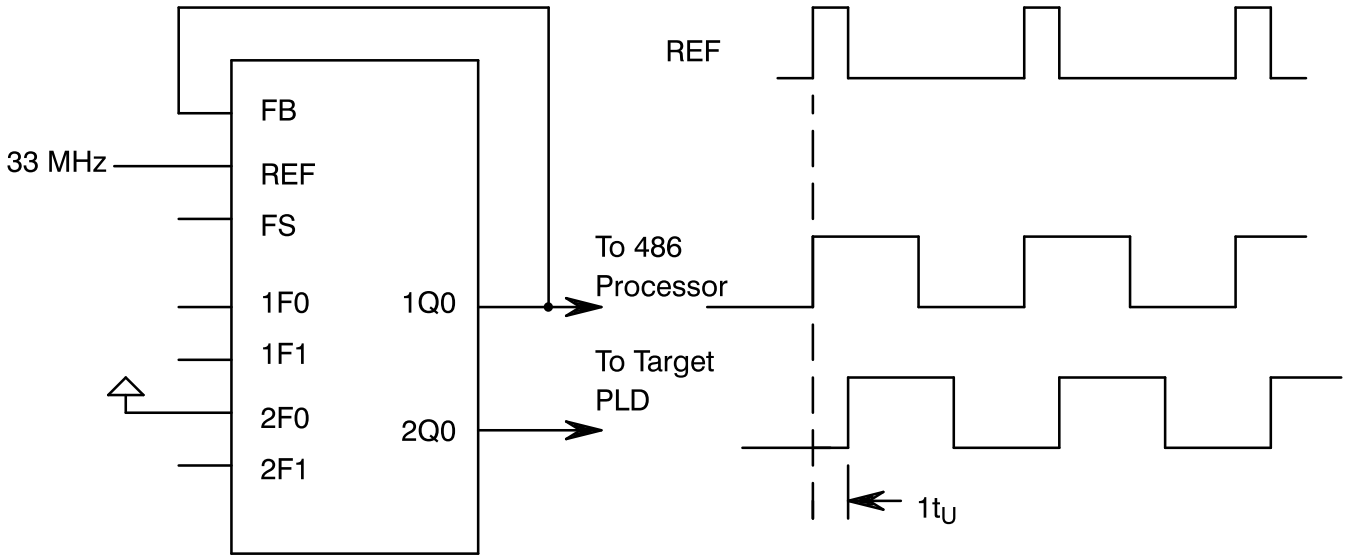

Figure 2. Timing Violation Solution

problem. Essentially, the designer used RoboClock to "move the clock", adjusting the phase of the 33-MHz clock input to the target PLD. As is shown, the 1F0 and 1F1 control inputs are allowed to "float" (the MID logic state), the resultant 1Q0 output is a buffered 33-MHz clock phase-aligned with the 33-MHz reference input. The 2F0 input is tied HIGH and the 2F1 input is allowed to float, configuring the 2Q0 output to be delayed by one timing unit (t_U), thus yielding 1.2 ns of margin to the circuit. The delayed 2Q0 output is then routed to the clock input of the target PLD, the 1Q0 clock is distributed to the clock input of the 486 processor.

In summary, the buffered clock signal, coincident with the reference clock, is distributed to the 486 processor, and a delayed clock is distributed to the target PLD, allowing the PLD's set-up time to be satisfied. An alternate solution would have been to distribute an advanced clock to the 486 processor, and distributing a nominal clock to the target PLD. Either solution may be implemented with RoboClock.

Note that the reference input as shown in *Figure 2,* need not be a "50-50" duty cycle signal in order for RoboClock to output a guaranteed 50−50 duty cycle clock. Duty cycle requirements are increasingly important for contemporary processors.

RoboClock as a Zero Propagation Delay Buffer

Clock speeds of 33 MHz and higher have become the norm in the modern design environment. Increasing clock rates mean decreasing clock periods, resulting in less "processing time" between rising clock edges. Conventional clock buffering methodology is no longer adequate for these applications because the propagation delay through a traditional "244" buffer is a significant portion of the clock period.

Even "high-performance, low skew" buffers suffer from some finite amount of propagation delay. This inherent delay translates into processing time lost to the designer. Fortunately, designers can benefit from PLL-based clock buffers, which are able to offer zero propagation delay. The following example is based on an actual implementation at a major manufacturer of ATM Network Adaptor cards.

Figure 3 shows RoboClock configured to operate in its most elegant mode, that of a zero propagation delay clock buffer. It is contrasted against a conventional clock buffering solution, such as a "244" buffer, which has inherent, performance degrading finite propagation delay. The specific version of RoboClock shown is the CY7B9910, which is a functional subset of the original, more fully featured, CY7B991. The 7B9910 or "Robo Jr." features an identical PLL core as its parent, and thus the identical excellent low-skew characteristics between buffered outputs. Robo Jr. was designed to be exclusively a low-skew, zero propagation delay clock buffer. It therefore lacks the previously described clock phase configurability that enables RoboClock to negate clock skew.

RoboClock as a Universal Clock Multiplier

Figure 4 shows RoboClock's ability to synthesize, with the addition of an external counter, any integer multiple of the reference frequency, up to a limit of 80 MHz. RoboClock has the built-in ability to multiply the reference clock by two and by four. Use of an external counter greatly expands this ability. This example is based on an actual design implemented by a major telecommunications manufacturer.

This frequency synthesis/multiplication is accomplished, as shown in this example, by feeding the terminal count output of a divide-by-three counter into the Feedback (FB) input of RoboClock. The relative phase difference present at the REF and FB inputs causes the internal PLL to adjust its output until these two inputs are phase aligned. This results in the outputs tripling in speed, from 20 MHz to 60 MHz. Additionally, the 60-MHz 4Q0 output is shown to be inverted. This was accomplished by setting the 4F0 and 4F1 control inputs as "High, High", the inverting configuration. Alternatively, the 60-MHz 4Q0 output could have been configured in a phase-adjusted mode—pushed forward several timing units or likewise pulled back. Thus the use of the external counter to accomplish clock multiplica-

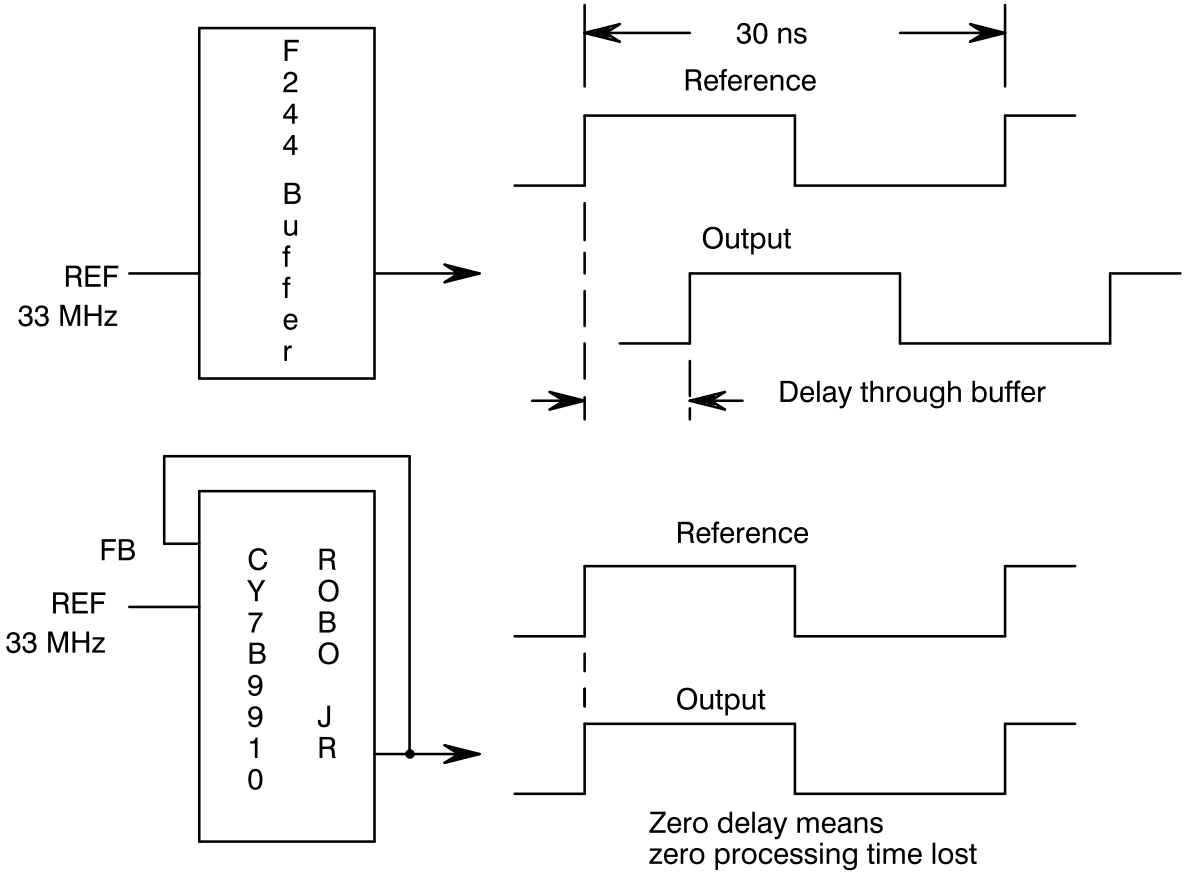

Figure 3. Zero Propagation Delay Buffer

tion in no way inhibits the normal features of RoboClock.

As shown in *Figure 4*, the multiplied outputs are slightly skewed from the Reference input. This phase difference results from the inherent "clock to output" delay of the external counter. Any counter will have some finite delay, which will manifest itself in this manner.

Should this skew be considered detrimental to the application, it can easily be eliminated on the 2Q, 3Q and 4Q outputs by adjusting the respective phases of the outputs. By the nature of the application, this skew will always be present between the 1Q output and the REF/FB input.

Gated RoboClock

From time to time, design requirements necessitate the "gating" of clock signals. Special care must be taken whenever this is done in order to prevent intermittent glitching of the distributed clock, the result of enabling and disabling. Clock glitches may result in minimum pulse width violations upon registers present on the circuit board. The application shown is based upon similar implementations in both an ESCON (Enterprise System CONnection) product as well as a high resolution graphics subsystem employed in a virtual reality product.

Should the output of RoboClock need to be gated, or three-stated, a viable method is shown in *Figure 5*. The CYBUS3384 is essentially a zero propagation delay (125 ps worst case) three-state buffer. The upper half of the CYBUS3384 is continually enabled, allowing a continuous wave form into the FB input of RoboClock. The output of the lower half of the CYBUS3384 is the gated clock, present when the CYBUS3384's $\overline{EN}$ signal is asserted (active LOW) and three-stated otherwise. The output enabling scheme shown, which has a register sampling the $\overline{EN}$ input upon the falling edge of the 1Q

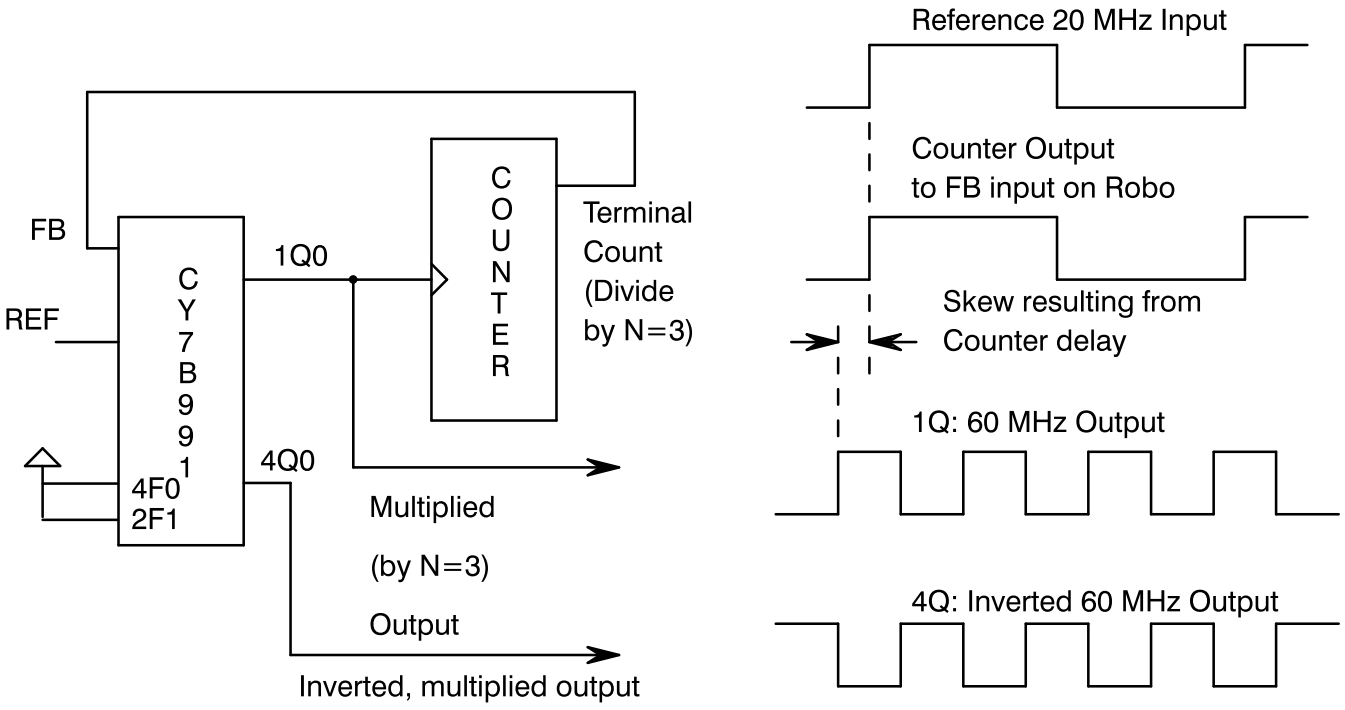

Figure 4. Universal Clock Multiplier

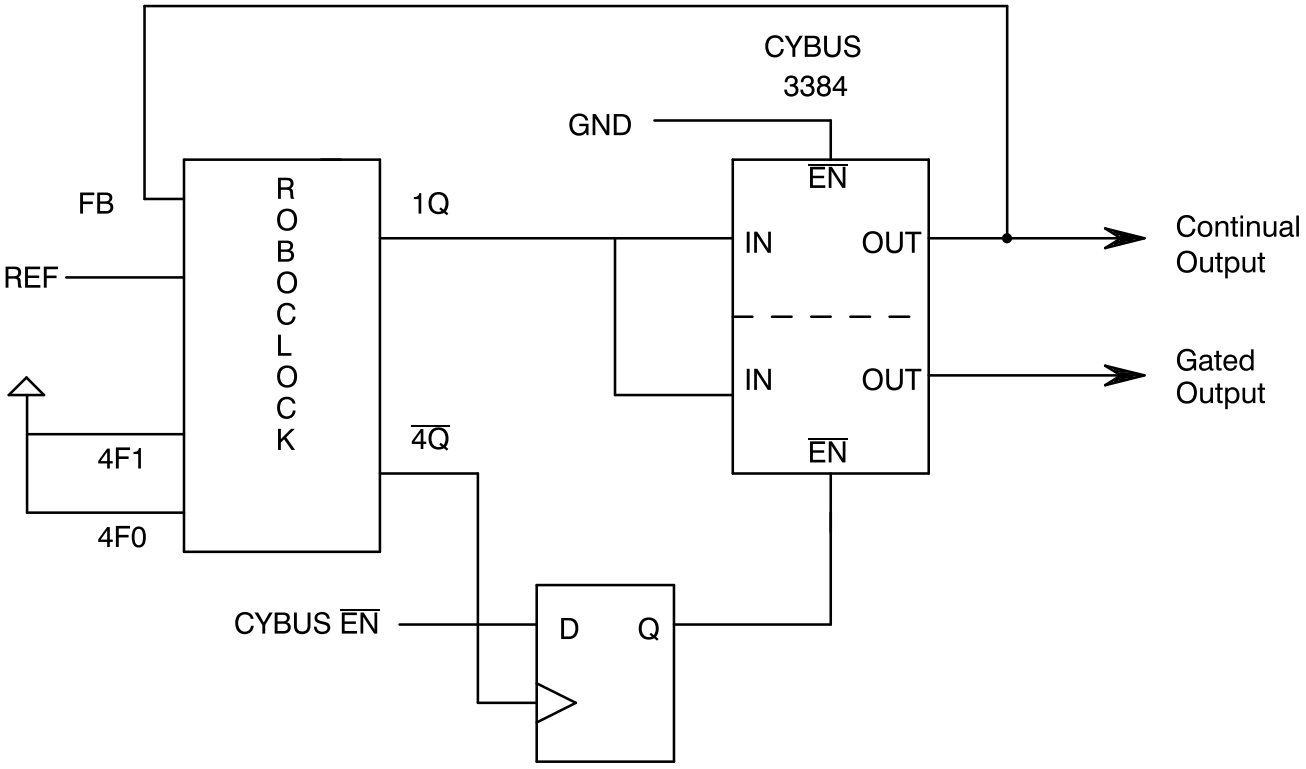

Figure 5. Gated RoboClock

clock (the 4Q output has been inverted, and thus a rising edge on 4Q corresponds to a falling edge on 1Q) guarantees that the gated clock output will never "glitch."

Continually Phase Adjusted Clock Source

As a result of the flexible configuration options within RoboClock, it is possible to achieve virtually 360 degrees of phase adjustment, allowing "placement" of clock edges throughout the period of a reference wave form. This functionality has been implemented in a telecommunications network analysis system used by Regional Bell Operating Companies (RBOCs) and is depicted in *Figures 2* and *7*.

In this application, a 33-MHz reference signal is dynamically phase adjusted by writing different values into a CMOS output level register, which in turn feeds the 3F and 4F RoboClock inputs. Note that the register used must have CMOS outputs, i.e., they must go "rail-to-rail" in order to satisfy the input level requirements of RoboClock. The register must also be capable of being three-stated, so that it can put the 3F and 4F RoboClock inputs into the "MID" state.

The application shown offers the designer the ability to subdivide the 30 ns period into thirteen slices, 2.4 ns apart. Each configuration of the 3F and 4F inputs corresponds to a different phase adjustment of the buffered clock, relative to the reference clock.

Conclusion

Today's high-performance design environments require the design engineer to work with and distribute high-speed clocks. By their nature these high-frequency clocks make the designer's task difficult. When these clocks have to be distributed over even relatively short distances, the effects of clock skew can make the designer's job impossible. The RoboClock family offers the design engineer opportunities to overcome a myriad of design challenges. Its ability to manipulate clock waveforms, and to counteract the effects of clock skew make it an integral part of the contemporary designer's repertoire.

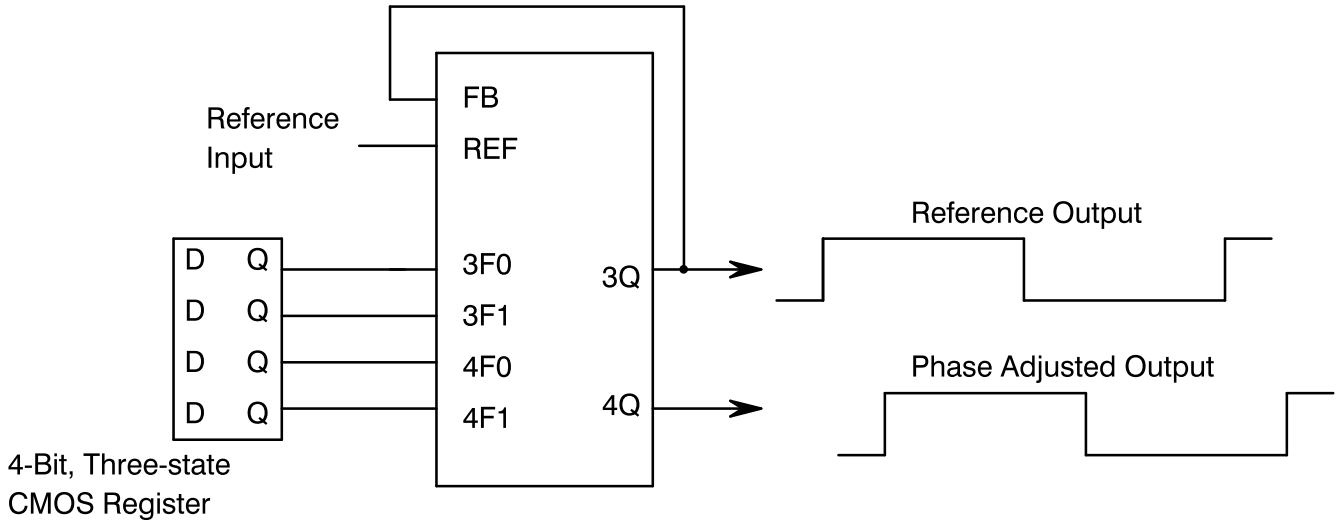

Figure 6. Continual Phase Adjustment

Control Inputs			
3F0	3F1	4F0	4F1
MID	MID	HIGH	MID
MID	MID	LOW	HIGH
MID	MID	MID	HIGH
MID	LOW	HIGH	MID
MID	LOW	LOW	HIGH
MID	LOW	MID	HIGH
MID	MID	LOW	MID
MID	MID	HIGH	LOW
MID	MID	MID	LOW
MID	HIGH	LOW	MID
MID	HIGH	HIGH	LOW
MID	HIGH	MID	LOW

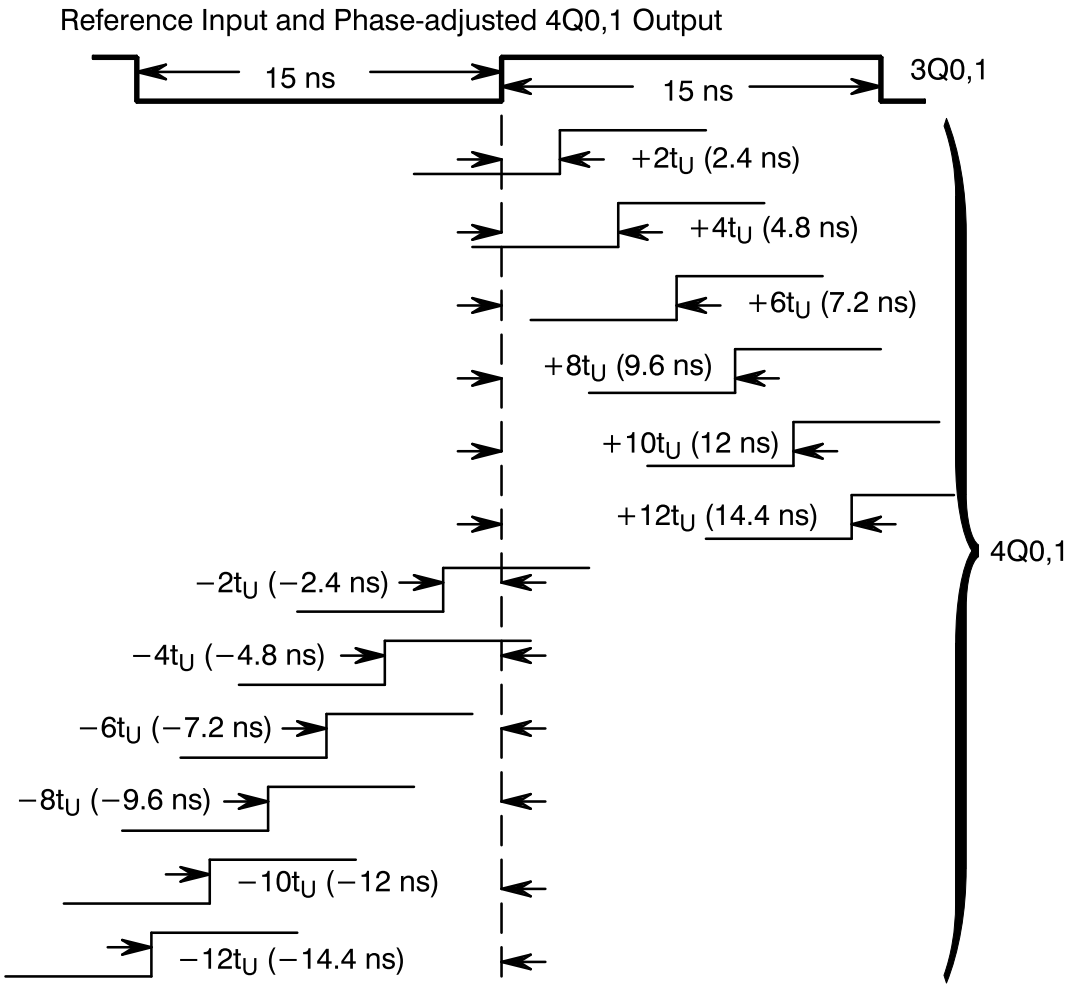

Figure 7. Continual Phase Adjustment