System implements digital-clock modulation

Dan Doberstein, DKD Instruments, Nipomo, CA - October 28, 2004

In spread-spectrum and direct-sequence receivers, it's often necessary to change the frequency of the clock oscillator, and thus the spreading-code clock, to lock the receiver's reference pseudorandom-noise code to the incoming code. The original design used a VCXO (voltage-controlled crystal oscillator) for this function, but this revised design implements an all-digital method that provided additional functions, using a midlevel CPLD from Altera. A commercially available part, the 74HC297, provides the function. This part is an all-digital PLL chip that most engineers have never heard of. The 74HC297 can digitally modulate an applied clock. It performs this task by adding or subtracting a pulse to or from a divided-down clock.

Upon examining the functions of this part, two issues emerge. First, it divides the input clock by two before modulating it. Second, the modulation method inserts and subtracts pulses. Although the pulse-addition/subtraction issue is acceptable, the divide-by-two function reduces the phase/frequency-control resolution. With the inspiration of the 74HC297 in mind, we sought a method that had finer resolution in phase and frequency for the same applied clock frequency. At first, we tried discrete counters using a digital switch to select different divide ratios. This approach worked at low clock frequencies but failed as clock speed increased because of propagation-delay effects at the switch. Eventually, we devised a method that used a long shift register and a tap-selection switch to implement a variable clock divider/phase modulator. Figure 1 shows the block diagram of the system.
Figure 1 This scheme allows you to change the rate of the spreading-code clock in spread-spectrum applications.

You load the 21-bit shift register with a single one and the rest zeros. A switch-selectable tap point feeds back to the shift register’s input. In effect, a single one now goes around and around at a frequency, $F_{OUT}$, determined by the tap point and the applied clock, $F_{CLK}$. The switch function allows selecting one of three divide ratios: 19, 20, or 21. The choice of the tap points is a function of the target application's needs, although you can choose other tap points. In operation, if the system detects an Advance pulse, the switch selects the divide-by-19 for one $F_{OUT}$ cycle. If the system detects a Retard pulse, the switch selects divide-by-21 for one $F_{OUT}$ cycle. After every Retard or Advance pulse, the system returns the selection switch to the divide-by-20 position. The single one, as it shifts through the register, effects the setting and resetting of the switch. At first, it appears that this system will modulate the $F_{CLK}$ input in an FM-like fashion. But, on second thought, it also acts as a phase modulator. On a single Advance pulse, approximately one-twentieth of an output cycle is subtracted, and a single Retard pulse adds approximately one-twentieth of an output cycle. If the Advance or Retard pulses are continuous streams at frequency $F_{MOD}$, the output frequency is $F_{OUT} = (F_{CLK} \pm F_{MOD})/N$, where $N=20$ in this case, and the sign is positive for Advance pulses and negative for Retard pulses.

So, this modulator can do both PM and FM on the applied clock, $F_{CLK}$. The tap numbers this design uses reflect those that fit the target application. The target application needs approximately 1-MHz final output frequency with a phase resolution of one-twentieth of a cycle, or the equivalent of 50 µsec in the time domain. If you need more phase resolution, increase the shift register's length. For example, a 101-stage shift register with tap points at 99, 100, and 101 has a phase resolution of approximately 3.6°. The advantages of the shift-register approach are speed and scalability. You can clock shift registers at extremely high speed. Additionally, the shift-
register approach automatically provides you with clock signals that are exact advanced or delayed versions of the primary tap used for $F_{\text{OUT}}$. This feature is useful in creating precise phase-related signals. The feature also provides an advantage in the control of the switch function, so that you can manage propagation-delay issues.

**Figure 2** shows the schematic in Altera's Max+Plus II design software. This software allows the use of standard logic-gate symbols in schematic format to create a programming file for the device. Although this design uses a CPLD, a discrete version using digital-logic ICs, as the schematic shows, should also work. The $F_{\text{CLK}}$ input connects to all the shift register's clock inputs. A Reset function initializes the registers with a single one and the rest zeros. Using a single flip-flop, $F_{\text{CLK}}$ samples the asynchronous reset signal to ensure synchronous operation. Three sets of flip-flops prepare the Advance/Retard signal for use by the selection switch. From the right-hand side of **Figure 2**, the first set of flip-flops on the Advance and Retard inputs ensures that these signals are synchronous with the final output frequency, $F_{\text{OUT}}$, and stores them for the next cycle of $F_{\text{OUT}}$. We assumed that these signals would be asynchronous with the input clock, $F_{\text{CLK}}$. The NAND gate ensures that both of these flip-flops clear if Advance/Retard pulses arrive simultaneously, an illegal input condition.

![Figure 2](image-url)

**Figure 2** Altera’s software allows you to enter standard logic-gate symbols to create a programmed CPLD.

The second set of flip-flops captures the rising edges of the Advance/Retard inputs. Subsequent rising edges are ignored for a complete cycle of $F_{\text{OUT}}$. This set of flip-flops holds the state of the selection switch for the next cycle of $F_{\text{OUT}}$. After every cycle of $F_{\text{OUT}}$, these flip-flops reset to the divide-by-20 state. The output of the third set of flip-flops controls the AND/OR switch logic, which selects the divide ratio for the current cycle. As the single logic one shifts to the right, it first latches the selected switch position into the last set of flip-flops. This selection could be divide-by-19, -20, or -21, depending on the Advance/Retard inputs. The AND/OR gates now can select which tap point connects to the shift register's input for that cycle of $F_{\text{OUT}}$. After one more shift, the single logic one clears the second set of flip-flops, thus returning to the divide-by-20 state.

A propagation delay is inherent in the Advance/Retard control inputs. If you apply an Advance/Retard pulse, it will not be applied to the selection of the tap point until the next cycle of $F_{\text{OUT}}$. We programmed the modulator into an Altera EPM7128-10 (10-nsec) device, and used an input-clock frequency, $F_{\text{CLK}}$, of 20 MHz. When the divide-select switch is in the divide-by-20 state, it is easy to compute the frequency output; it’s just the input clock divided by 20. For an N-bit system, it would be the input clock divided by N. But how do you derive a general formula for $F_{\text{OUT}}$ if Advance/Retard
pulses arrive at a rate of $F_{\text{MOD}}$. Whenever an Advance or Return pulse is processed, a fixed amount of time is added or subtracted to the time between $F_{\text{OUT}}$ pulses. You need an expression for the equivalent amount of added or subtracted phase—in other words, the phase step.

One complete cycle of $F_{\text{OUT}}$ with no Advance/Retard modulation takes $N/F_{\text{CLK}}$ seconds. This interval is just the period of the output with no modulation. If you add or subtract one clock period, how much phase does this represent with respect to $F_{\text{OUT}}=N/F_{\text{CLK}}$? In terms of the fraction of time that adding or subtracting one clock period from the nominal output cycle, you can write: Fraction of one output cycle per Advance/Retard pulse=$(\text{clock period})/(\text{nominal-output period})=(1/F_{\text{CLK}})(F_{\text{CLK}}/N)\text{ cycles}=1/N$ cycles.

Converting to radians and defining this step as the phase step or $\Phi_{\text{STEP}}$, $\Phi_{\text{STEP}}=2\pi/N$ radians. Therefore, every time an Advance or Retard pulse is processed, the phase of the output changes by $\pm 2\pi/N$ radians. You can now use the fact that frequency is the time derivative of phase to derive the formula for $F_{\text{OUT}}$. Figure 3 shows a time plot of phase of the output $F_{\text{OUT}}$ for the three possible commands: Advance, Static (divide-by-N), and Retard. In Figure 3, at first phase is added, no change occurs in phase, and then phase is subtracted. The frequency is not changed where the slope is zero, and is equal to $F_{\text{CLK}}/N$. At the stair-step portions of the plot, you can approximate the slope as $\pm 2\pi/NT_{\text{MOD}}$. You can interpret this figure as the change in the output frequency (in radians). To convert to cycles, divide by $2\pi$, which gives the change in output frequency as $\pm F_{\text{MOD}}/N$ for Advance or Retard pulses arriving at a rate of $F_{\text{MOD}}$.

![Figure 3](image)

**Figure 3** The stair-step portions of this graphic show where phase is added to or subtracted from $F_{\text{OUT}}$.

**PDF Version**

Also see:

- Stepper-motor motion controller and driver fit into a CPLD/FPGA
- Spread-spectrum clocking in PCI Express