

# Frequency Fine Tuning and Clock Dithering Using Actel FPGA Devices

## Introduction

The PLLs embedded within Actel FPGAs offer a variety of divider and multiplier blocks for frequency synthesis. These embedded dividers and multipliers can be configured dynamically during operation to change the PLL output frequency while the reference clock remains unchanged. However, the changes in the PLL frequency using the embedded divider and multiplier blocks can be coarse enough to cause the PLL to lose lock temporarily while gearing from one frequency to another.

Some applications are extremely sensitive to the frequency changes. In these cases it is vital for the PLL to remain locked while transitioning from one frequency to another. The reference design, presented in this application note, uses the external feedback feature of Actel PLLs and places a fully configurable frequency tuning/dithering block inside the PLL feedback path. The tuning/dithering block ramps up/down the PLL output frequency with configurable frequency steps and configurable time interval between the steps. [Figure 1](#) shows a block diagram of top level implementation of the design.

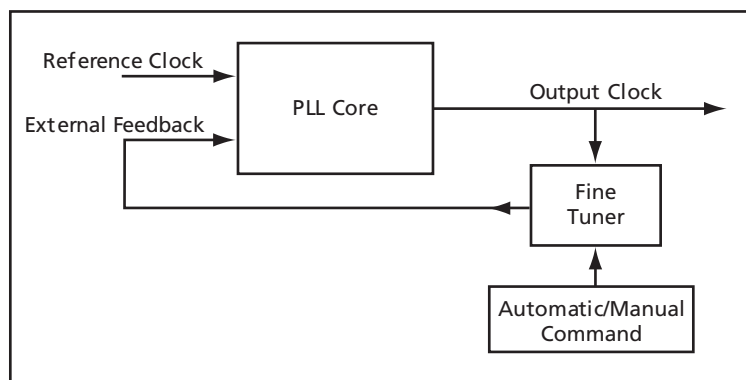


Figure 1 • Top Level implementation of PLL Fine Tuning/Clock Dithering Design

As shown in [Figure 1](#), the internal feedback path of the PLL is opened and the Fine Tuner block is placed within the external feedback loop. The Fine Tuner block is controlled by a command block which instructs the tuner block to increase/decrease the PLL frequency by one step. The command block can be designed to either automatically increase/decrease frequency to the pre-defined values within a pre-defined time frame or honor manual commands to increase/decrease the frequency by just one step.

The following sections of this document describe each building block of [Figure 1](#).

## PLL Core with External Feedback

Actel recommends that engineers use ACTgen to create the PLL macro with the required configuration. From within the ACTgen GUI, users can select to use an external feedback. By selecting the external feedback option, an additional input port will be added to the PLL high level macro which is the entry point to the external feedback loop. Depending on the selected FPGA family, this port is named FB or EXTFB.

The following is an example of VHDL entity of a ProASIC<sup>PLUS</sup> PLL with external feedback input:

```
entity APA_PLL is
  port (EXTFB, CLK : in std_logic;
        LOCK, GLB  : out std_logic);
end APA_PLL;
```

In Axcelerator FPGA devices, the external feedback port of the PLL can be driven by either internal nets or I/O pins. In ProASIC<sup>PLUS</sup> devices, the external feedback of the PLL can only be driven by the pins designated as GLMX (two per die). Since the tuner/dithering block is implemented inside the FPGA, the user can exploit a bidirectional I/O macro at the GLMX pin to avoid routing the PLL feedback signal in and out of the FPGA. This solution is shown in Figure 2. The used GLMX pin of the FPGA must be left floating at the board-level.

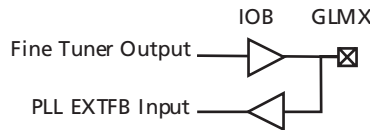


Figure 2 • PLL External Feedback Connection in ProASIC<sup>PLUS</sup> Devices

## Fine Tuner

Fine tuner block is the most important block of this reference design. In order to have a good understanding of the finer tuner block design and define proper tuner configuration parameters for each application, it is important to understand the basic concept of the functionality of this block.

## Functionality

The functionality of the tuner block is based on the basics of PLL operation. After the PLL is locked onto the input frequency, any changes to the inputs of the PLL phase detector (either change in input clock or in feedback clock) will cause the PLL VCO to shift into a different operating point (phase or frequency) to compensate for the phase difference at the inputs of the phase detector. The fine tuner block is placed in the feedback loop of the PLL and changes the phase/frequency of the signal sent to the phase detector feedback input. This causes the VCO to operate faster or slower to compensate for the changes induced by the tuner block.

The tuner block functions as a configurable mask where it receives the PLL output frequency, masks out one cycle out of every defined number of cycles, and passes the clock to the PLL feedback input.

Figure 3 shows an example of the tuner block input and output.

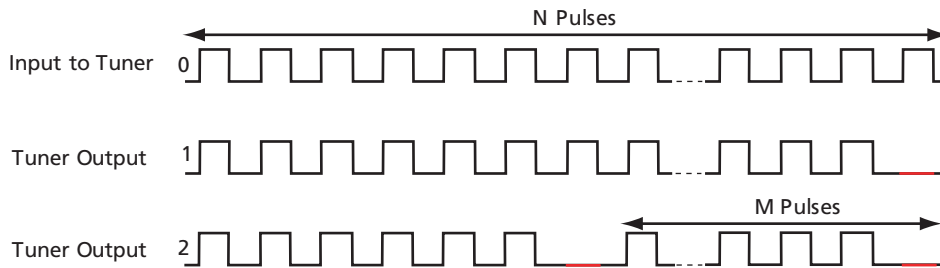


Figure 3 • Example of Input and Output of Tuner Block

In Figure 3, Signal 0 represents the PLL output that is applied to the input to the tuner block. It can also be the tuner output if the tuner block does not make any changes to the PLL output frequency as defined in ACTgen. Signal 1 represents the moment in which the tuner block masks out one pulse out of every N pulses. If the default output frequency of PLL is F, feeding Signal 1 to the feedback input of the PLL causes the VCO to operate at the value shown in EQ 1.

$$F \times \frac{N+1}{N}$$

EQ 1

Similar to Signal 1, Signal 2 masks one pulse out of each M pulses. Feeding this signal to the PLL feedback input will cause the VCO of the PLL to operate at the value shown in EQ 2.

$$F \times \frac{M+1}{M}$$

EQ 2

Consequently the step sizes moving from Signal 0 to Signal 1 and from Signal 1 to Signal 2 is shown in EQ 3 and EQ 4 respectively.

$$\frac{1}{N}F$$

EQ 3

$$\left(\frac{1}{M} - \frac{1}{N}\right)F$$

EQ 4

A different variation of the tuner functionality can be masking q pulses out of N pulses, where N is fixed and q is varying from 0 to a defined number (e.g., Q) between the steps. However, in this implementation, when q > 1 the eliminated pulses should be distributed evenly throughout the N pulses to avoid introducing a big jitter noise to the PLL.

## Implementation

Figure 4 shows a simplified block diagram of the **Fine Tuner** implementation.

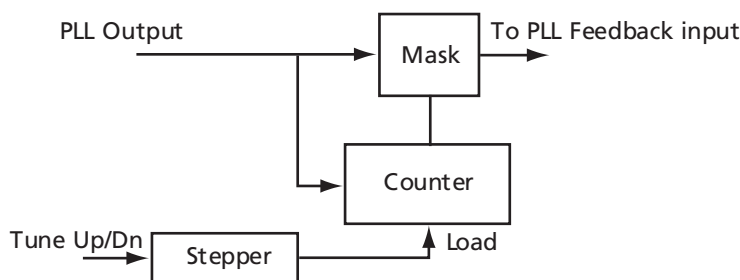


Figure 4 • Simplified Block Diagram of Fine Tuner Implementation

In Figure 4, the Stepper is a state machine (or a memory block) which stores all the steps and walks through them and loads them into the counter with each tune up/down command. The Counter is a count up or down module that is driven by the PLL output clock. After Counter completes counting the number of clock cycles loaded by Stepper (e.g. N), it asserts a mask pulse for one clock cycle to mask out one cycle of the PLL clock output in the PLL feedback loop. When the Stepper changes the load value based on the incoming step command, the counter will load the new value only after the current count cycle is complete. This is done to avoid unwanted glitches and additional jitter input to the PLL.

To implement a high speed counter, a fast increment or decrement block (generated by ACTgen) can be used instead of conventional counter blocks to implement the counter module. The Mask block can be a gating module such as an AND gate.

## Fine Tuner HDL Design

This section introduces the HDL design of the top tuner block described in the previous section along with simulation results. The design is coded in both VHDL and Verilog HDL format. The HDL source code can be found at the following links on the Actel website:

Verilog: [http://www.actel.com/documents/Clock\\_Dithering\\_Verilog.zip](http://www.actel.com/documents/Clock_Dithering_Verilog.zip)

VHDL: [http://www.actel.com/documents/Clock\\_Dithering\\_VHDL.zip](http://www.actel.com/documents/Clock_Dithering_VHDL.zip)

### Design Interface

#### *The Fine Tuner Block*

The HDL design ports are as follows:

**Pllout\_clk:** Input. Connects to the output of the PLL.

**Pll\_extfb\_in:** Output. Connects to the PLL external feedback input.

**Step\_up:** Input command pulse to increase the PLL frequency by one step. The command pulse must be one clock cycle wide.

**Step\_dn:** Input command pulse to decrease the PLL frequency by one step. Similar to step up command, this pulse must be one clock cycle wide.

In this reference design, if step\_up command will have priority over step\_dn if both signals are asserted at the same time.

**Rst\_n:** Input global reset. Resetting the tuner block will reset the PLL output frequency to a user-defined default frequency.

### Generics and Parameters

The top Fine Tuner HDL code contains three generic/parameter values that must be set by the user according to each specific design.

These generics/parameters are:

**STEP\_VALUE\_MIN:** Minimum number of pulses in which one clock pulse is masked. This value corresponds to the maximum clock frequency that can be achieved by the tuner block

**STEP\_VALUE\_MAX:** Maximum number of pulses in which one clock pulse is masked. This value corresponds to the minimum clock frequency that can be achieved by the tuner block.

**INITIATE\_VALUE:** Number of pulses in which one clock pulse is masked at reset time which defines the initial clock frequency at reset.

**WIDTH:** The width of the counter. This can be defined using [EQ 5](#).

$$2^{WIDTH-1} < STEP\_VALUE\_MAX < 2^{WIDTH}$$

EQ 5

The following example illustrates how the generics of the design are defined. Assume that the PLL output frequency should be tuned up and down between 50.1 MHz and 50.5 MHz. The clock input PLL is set to 50MHz and all the internal divider and multipliers blocks are set to one.

STEP\_VALUE\_MAX can be calculated based on the upper side of the tuning range (maximum frequency), as shown in [EQ 6](#).

$$\frac{STEP\_VALUE\_MIN + 1}{STEP\_VALUE\_MIN} \times 50MHz = 50.5MHz \Rightarrow STEP\_VALUE\_MIN = 100$$

EQ 6

Similarly, STEP\_VALUE\_MAX can be calculated based on the lower side of the tuning range, as in EQ 7.

$$\frac{\text{STEP\_VALUE\_MAX} + 1}{\text{STEP\_VALUE\_MAX}} \times 50 = 50.1 \Rightarrow \text{STEP\_VALUE\_MAX} = 500$$

EQ 7

Since by design, the maximum step size occurs at the last step when the tuner frequency reaches its maximum range, the maximum step size is shown in EQ 8.

$$\left(\frac{101}{100} \times 50\text{MHz}\right) - \left(\frac{102}{101} \times 50\text{MHz}\right) = 0.005\text{MHz}$$

EQ 8

The INITIATE\_VALUE can be set to be anywhere between 100 and 500 per user's preference. In the example, WIDTH is set to be nine.

## Architecture Specific Implementation

Specific portions of the tuner block HDL design are architecture specific (gate level design) to achieve better performance in terms of clock frequency. Furthermore, these architecture-specific blocks can be optimized for each application to achieve even higher performance. More specifically, the architecture-specific blocks are the increment/decrement blocks used to build high performance counters. In the fine tuner design attached to this application note, ProASIC<sup>PLUS</sup> architecture is chosen to create high speed increment/decrement blocks from ACTgen. These increment/decrement blocks are in INCR.vhd (INCR.v) and DECR.vhd (DECR.v) files and instantiated in both stepper.vhd (stepper.v) and counter\_dec.vhd (counter\_dec.v). These blocks in the attached designs are behavioral HDL. However, to achieve a better performance, users can create a netlist version of these increment/decrement blocks using ACTgen (using the appropriate WIDTH value) and use them instead of the behavioral blocks attached to this document.

If any FPGA architecture other than ProASIC<sup>PLUS</sup> is used, the increment/decrement block netlists should be regenerated for that architecture.

Furthermore, the WIDTH value in the design should be set equal to the width of these increment/decrement netlist blocks, generated by the user.

The Fine Tuner design, as attached to this document, implemented within a wrapper block along with a PLL in an APA075 device, achieved more than 88MHz clock performance at worst-case condition when the WIDTH was set to 12 and fast increment/decrement blocks was generated by ACTgen and used in the design. This performance can be increased significantly by reducing the width of the counters if high step values are not used.

## Simulation Results

The top tuner design is simulated using impractical but easy to represent step values. Figure 5 shows a snapshot of the simulated waveforms.



Figure 5 • Simulation Waveform of Fine Tuner Design

As it was stated earlier and demonstrated by simulation, when the stepper block changes the step value in the middle of the count cycle, the counter block will not load the new step value unless the current count cycle is completed. If the stepper block changes the step value multiple times during a count cycle, the counter will load the latest step value after completing its current count cycle.

## Utilization

The Fine Tuner block is very small and therefore would not affect the rest of the user design during layout. The following is the Fine Tuner design utilization (with WIDTH set to 12) when implemented in ProASIC<sup>PLUS</sup> architecture:

Total number of used tiles: 294

Tiles used as register: 72

Tiles used as combinatorial gates: 222

Actel and the Actel logo are registered trademarks of Actel Corporation.  
All other trademarks are the property of their owners.



[www.actel.com](http://www.actel.com)

**Actel Corporation**

2061 Stierlin Court  
Mountain View, CA  
94043-4655 USA

**Phone** 650.318.4200  
**Fax** 650.318.4600

**Actel Europe Ltd.**

Dunlop House, Riverside Way  
Camberley, Surrey GU15 3YL  
United Kingdom

**Phone** +44 (0) 1276 401 450  
**Fax** +44 (0) 1276 401 490

**Actel Japan**

[www.jp.actel.com](http://www.jp.actel.com)

EXOS Ebisu Bldg. 4F  
1-24-14 Ebisu Shibuya-ku  
Tokyo 150 Japan

**Phone** +81.03.3445.7671  
**Fax** +81.03.3445.7668

**Actel Hong Kong**

[www.actel.com.cn](http://www.actel.com.cn)

Suite 2114, Two Pacific Place  
88 Queensway, Admiralty  
Hong Kong

**Phone** +852 2185 6460  
**Fax** +852 2185 6488