
Using Phase Locked Loop (PLLs) in DL6000 Family Devices

Introduction

Designers have used PLLs for many years (however, only recently with FPGAs) mainly to synchronize (sometimes referred to as deskewing the clock) the internal chip clock with an external clock. Synchronization allows chips in a system containing PLLs to have ‘virtually common’ clocks to each other. A PLL greatly reduces the clock latency of a device which in turn reduces the clock-to-out time of the device. Designers can also use PLLs to multiply the external clock to produce a clock to use within a chip, allowing for multiple clock domains on the same chip. The DL6000 includes two analog PLLs that can serve all of these applications. A standard PLL consists of a phase detector, loop filter, charge pump, and voltage controlled oscillator (VCO) elements, as shown in the block diagram Figure 1.

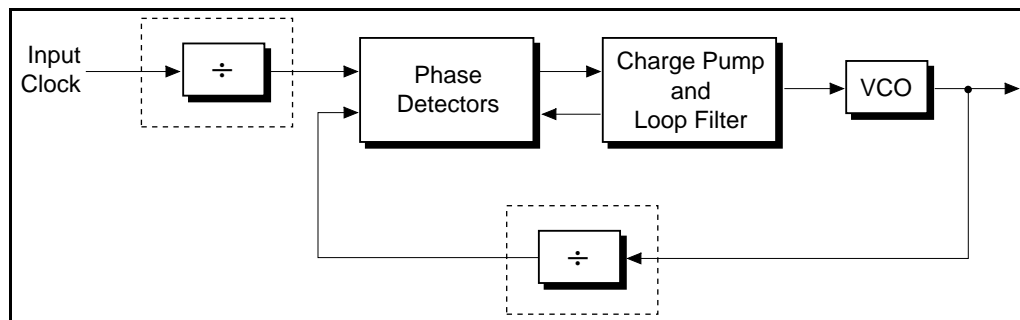


Figure 1 PLL Elements Block Diagram

The PLL compares the input clock frequency with the feedback frequency from the VCO and uses the phase detector, the loop filter, and the charge pump to change the frequency of the VCO. Once the input and feedback frequency are the same, the PLL is “locked” and maintains that frequency. DynaChip has added two divider circuits, shown in the boxed regions in Figure 1, to allow the PLL to multiply the input frequency which produces higher frequency elements of the input clock. If both dividers are set to 1, the PLL reverts to the basic structure. PLLs are made using both analog and “digital” design techniques. DynaChip's analog PLL is the preferred implementation because it requires much less silicon area and is based on a highly-proven technology.

DL6000 PLL Resources

Every DL6000 device contains two PLLs. In addition to locking the input global clock frequency, the DL6000 PLLs can multiply the input global clock frequency to produce higher multiples and lock to that multiple. Depending on the frequency, the global clock multiplication factor can be 8, 4, 3, 2, or 1. The PLL can generate frequencies as high as 205 MHz! An added benefit of using the PLL is the reduction in clock latency. In the DL6000 G speed grade, the clock latency is reduced from 5 ns to 0.9 ns, thereby reducing the effective

clock to out time by 4.1 ns. The input setup time is also reduced. In addition to the two PLLs, additional divider circuits were added to the quadrant clock circuits to produce additional lower frequencies synchronized to the global clock frequency. Each of the PLLs has a structure identical to the circuit illustrated in Figure 1. Figure 2 illustrates the PLL and the clock network, showing that the output of the PLL drives the global clock and four quadrant clocks. Each of the four quadrant clocks has its own divider circuit that allows division of the main clock frequency by either 1, 2, 3, 4, 6, or 8.

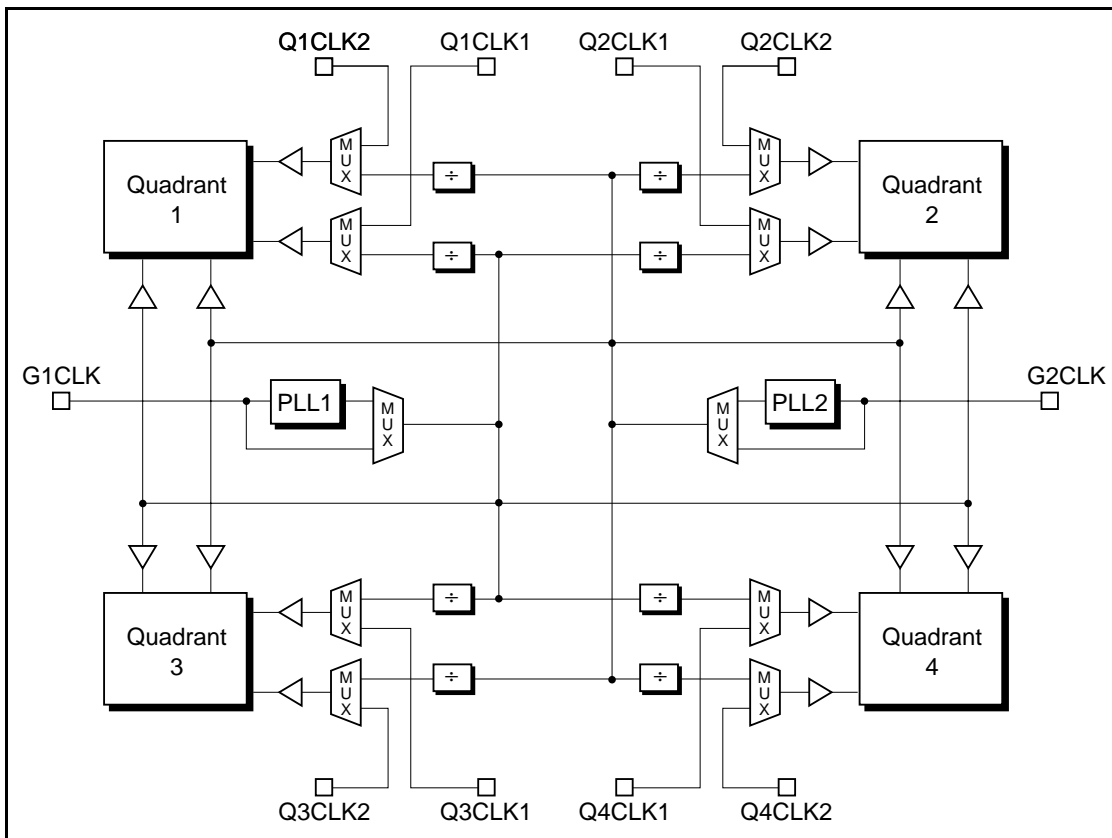


Figure 2 PLL and Clock Network

The two PLLs in the illustration are named PLL1 and PLL2. PLL2 has a slightly wider range than PLL1. In order to extend the locking frequency of either PLL, an external resistor for each PLL is required (a 14K Ω resistor is recommended). Two package pins are dedicated for this purpose. PLL1 can lock on any frequency between 95 MHz and 185 MHz without the use of an external resistor. An external resistor can lower the locking frequency of PLL1 down to 40 MHz. Without an external resistor, PLL2 can lock on any frequency between 90 MHz and 205 MHz. Using an external resistor extends this range down to 35 MHz. Between 90 MHz and 205 MHz, each PLL has a maximum jitter of 350 ps. Below 90 MHz, an external resistor is required and increases the jitter to 3% of the clock period. In order to meet these jitter specifications, the jitter of the input clock must be less than or equal to 100 ps.

Note: The frequency range refers only to the locking frequency, not to the input global clock frequency.

For example, if you are using a 38.75 MHz clock that you wish to multiply up to 155 MHz, either PLL1 or PLL2 will lock to 155 MHz since it is over 90 MHz. This also provides an effective lock on the 38.75 MHz. If you have only the 38.75 MHz and want to lock on that, an external resistor is required. Implementation details are given in examples 1 and 2, below.

DynaTool 2 Library Elements and Constraints

Before starting the implementation, it is useful to review the PLL1, PLL2, and QPLL macro library elements the DynaTool 2 software development system will use to define the PLL circuits.

- PLL1 defines the use of PLL1 along with global clock 1
- PLL2 defines the use of PLL2 along with global clock 2
- QPLL defines the use of any quadrant clock which is fed by a PLL

Along with using these macros, you must add PLL clock constraints to the constraint file to define the PLLs' use. An example below illustrates the use of these elements. Complete details on the elements' use are in the DynaTool 2 user's guide. You can add these macros either to the HDL description or to a schematic.

Note: The QPLL macro is not specific to any quadrant clock so you may use it as many as eight times in an application because there are eight quadrant clocks. You can use PLL1 and PLL2 only once.

Implementation

Implementing the PLLs is very straightforward. To start, you need to consider the following:

- Are one or two PLLs required?
- How many different frequencies are needed?
- What is the highest frequency?
- What is the lowest frequency?
- If a frequency under 90 MHz is required, can the design "tolerate" the use of an external resistor and the increased jitter?

Once you resolve these considerations, you can begin the implementation. The best way to convey the implementation process is to work through a couple of examples. The first example shows the use of the PLL driving a global clock and a quadrant clock. The second example shows the use of an external resistor.

Example: 1

This example shows a requirement for two clock domains. On one clock domain, a 38.5 MHz clock drives 300 logic blocks. The second clock runs at 4x the 38.75 MHz clock and drives 400 logic blocks. The only clock available on the pads runs at 38.75 MHz. A diagram of this circuit is shown in Figure 3.

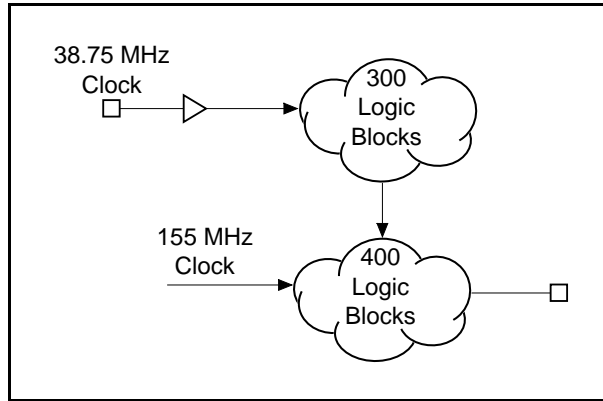


Figure 3 Clock Domain Circuit

Since only one PLL is needed, PLL1 is chosen though PLL2 could also have been chosen. PLL1 is driven with the 38.75 MHz clock and the frequency is multiplied by four in order to generate the 155 MHz clock. Once multiplied by four, the 155 MHz clock will drive global clock 1. This frequency is divided by four to reproduce the original 38.5 MHz and drive that signal onto quadrant 1 and quadrant 2. (Since the 38.75 MHz clock drives 300 blocks, two quadrants are required.) Quadrant 1 and 2 are chosen randomly. This data needs to be transferred to DynaTool 2 to produce the proper effect. DynaTool will need to see PLL1 and QPLL added to the design. Figure 4 shows how these two macros will be arranged in a schematic.

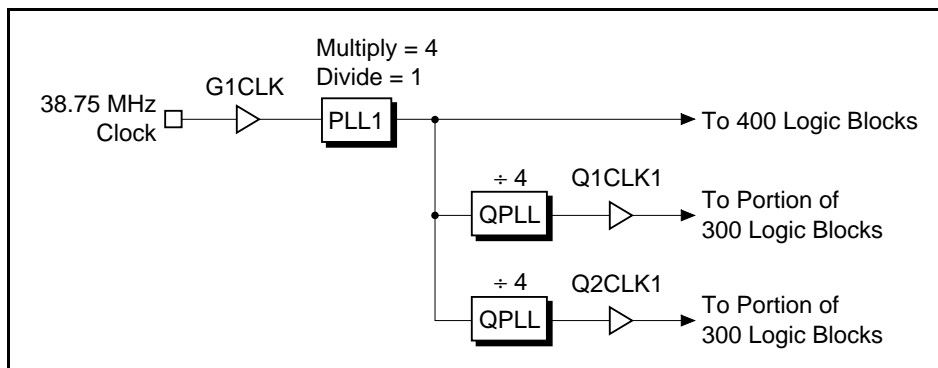


Figure 4 Macro Arrangement in the Schematic

Since PLL1 can be fed only by global clock 1, the library macro element G1CLK is connected to the pad. The macro for global clock 1 feeds the macro PLL1, which adds the PLL to the circuit and allows it to multiply the input signal by four. The output of PLL1 feeds those circuits the 155 MHz clock will drive. Also, the output of PLL1 feeds 2 QPLL library macro elements which in turn feed the macro for quadrant clock 1, clock 1 and quadrant clock 2, and

clock 1. The two quadrant clock macros feed those circuits to be driven at the 38.75 MHz frequency. Instead of implementing these macros in a schematic, you can also implement them in an HDL. Figure 5 shows the code necessary to implement the clock and PLL library elements in VHDL (you can also use Verilog).

```

uclk1: G1CLK port map (a=>clk, y=>in1);
upll1: PLL1 port map (a=>in1, y=>in2);
uqpll1: QPLL port map (a=>in2, y=>iqpll1);
uqpll2: QPLL port map (a=>in2, y=>iqpll2);
qclk1: Q1CLK1 port map (a=>iqpll1, y=>iqck1);
qclk2: Q2CLK1 port map (a=>iqpll2, y=>iqck2);

```

Figure 5 VHDL Code to Implement the PLL Circuit for Example 1

You should include this code along with all other top-level library elements (inputs, outputs, global set/reset, and I/Os), then input the circuit to DynaTool 2 for implementation. In addition to creating the circuit description, you will also need to add three lines, shown in Figure 6, to the constraint file to inform DynaTool 2 exactly how PLL1 and the 2 quadrant clocks should perform. The first line shows how PLL1 performs its multiply function. The next two lines show how the two quadrant clocks divide the clock by four. You can use any quadrant divider circuit to divide the main clock by either 1, 2, 3, 4, 6, or 8. The frequency of the global clock does not affect this value.

```

PLL1 exmp.upll1 -MULTIPLY 4 -DIVIDE 1 4X_CLK
QPLL exmp.uqpll1 -DIVIDE 4 1X_QCLK1
QPLL exmp.uqpll2 -DIVIDE 4 1X_QCLK2

```

Figure 6 Constraint File Additions for Example 1

Example: 2

The circuit shown in Figure 7 illustrates this example.

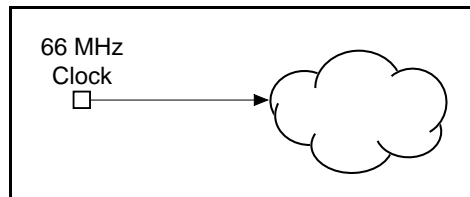


Figure 7 Example Clock Circuit

The input frequency of 66 MHz is what needs to be locked to; for example, to provide a very fast clock to out by using the PLL to reduce the latency of the clock. Since 66 MHz is below 90 MHz, two alternatives are available for locking the frequency. One method is to take the 66 MHz, multiply it by two to produce a 132 MHz clock on a global clock, then divide that down by two to produce the 66 MHz to drive onto all four quadrant clocks. This is a viable alternative, but in this case is rejected to save the power used to drive the global clock grid. Because the frequency is below 90 MHz, adding an external resistor (a 14K Ω resistor is

recommended) is required as well as checking to see if the increased jitter is a problem. Since at 66 MHz the clock period is 15.152 ns, the jitter will be 3% of this value, which is 455 ps. This is an increase of 55 ps over that which would be the result if PLL were run at 173 MHz. The extra jitter, plus the external resistor, provide reasonable alternatives to saving power. For this example, PLL2 is chosen (PLL1 could also have been chosen). Since no quadrant clocks are being used, you need to specify only macro PLL2 in the schematic or HDL. The schematic of the circuit shown in Figure 8 is very similar to that of the previous example.

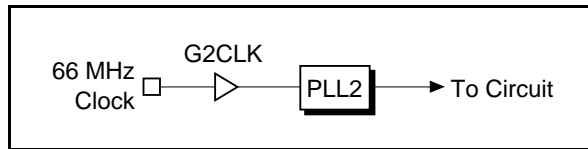


Figure 8 Example 2 Clock Circuit

The accompanying HDL code needed to implement the PLL circuit for this example is shown in Figure 9.

```
uclk1: G2CLK port map (a=>clk, y=>in1);
upll2: PLL2 port map (a=>in1, y=>in2);
```

Figure 9 VHDL Code to Implement the PLL Circuit for Example 2

You also need to add the constraint additions shown in Figure 10 to the constraint file so that DynaTool 2 makes the proper adjustments to the PLL2.

Since locking only to the 66 MHz frequency is desired, both the multiply and divide factors are '1.' You do not need to specify using a resistor pin since DynaTool 2 cannot use these two pins as I/O.

```
PLL1 exmp.upll1 -MULTIPLY 1-DIVIDE 1 1X_CLK
```

Figure 10 Constraint File Additions for Example 2

The DL6000 is the first FPGA to contain two fully-functional PLLs. Both PLLs not only can lock to any frequency between 40 MHz and 205 MHz (PLL2 can go as low as 35 MHz), they also can multiply a low frequency clock by 1, 2, 3, 4, 6, or 8. The eight-quadrant clocks also have the ability to divide the PLL output frequency by 1, 2, 3, 4, 6, or 8. Using PLLs requires that you follow a only few simple steps. Since the DL6000 PLLs are easy to implement and provide superior flexibility for clock management, you should seriously consider using them in your next design.