MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

Order this document
by AN1522/D

AN1522

Analog Phase Locked Loop for H4CPlus™, H4EPlus™ and M5C™ Series Arrays

Prepared by: Roy Jones
Edited by: Clarence Nakata
Application Specific Integrated Circuits Division, Chandler AZ

Table of Contents **Page**

1. Introduction

This application note describes the implementation and use of an analog phase locked loop, or APLL, which is available on two families of CMOS gate arrays offered by Motorola: the H4CPlus Series arrays, H4EPlus Series arrays and the M5C Series arrays.

Section 2 describes the various versions of the APLL which are offered as different library macros. This section also contains APLL performance data and signal descriptions, and shows the physical placement of the APLL on H4CPlus, H4EPlus and M5C arrays.

Section 3 describes how the APLL Verilog simulation model works and how it is used for "system-mode" simulations (as opposed to "option release" simulations).

Section 4 describes Motorola's strategy for testing H4CPlus, H4EPlus and M5C arrays that contain an APLL. A Motorola-internal test program is used to test the APLL itself, while user-supplied option release test vectors are used to test the remainder of the chip.

The Appendix shows a schematic of the test circuitry built into the APLL as well as a table of the various operating modes of this circuitry, which is controlled by the Motorola-internal test program.

2. Feature Description

Appendix D and Appendix E contain a comprehensive discussion and analysis of the use of Motorola's digital PLL (DPLL) to speed-up chip-to-chip data transfer by cancelling out on-chip clock network insertion delay. This analysis also applies to an analog PLL (APLL), which can be used for the same purpose.

Compared to the DPLL, the APLL can run faster, and has less in-lock phase error. The APLL also provides on-chip frequency synthesis, which allows a slower/quieter backplane clock frequency to be multiplied up to the desired on-chip clock frequency. The DPLL is not offered in H4CPlus, H4EPlus or M5C Series Arrays.

The following table lists the APLL macros available in H4CPlus, H4EPlus and M5C arrays.

Table 1 APLL Macros

Macro	Technology	Analog Power	FREF Input Type	N, Loop Divider
AP1	H4CPlus, H4EPlus	5 V	CMOS	1 - 4
APD1	H4CPlus, H4EPlus	5 V	PECL	1 - 4
AP2	H4CPlus, H4EPlus	5 V	CMOS	5 - 16
APD2	H4CPlus, H4EPlus	5 V	PECL	5 - 16
APL1	H4CPlus, H4EPlus	3.3 V	CMOS	1 - 4
APDL1	H4CPlus, H4EPlus	3.3 V	PECL	1 - 4
APL2	H4CPlus, H4EPlus	3.3 V	CMOS	5 - 16
APDL2	H4CPlus, H4EPlus	3.3 V	PECL	5 - 16
APL1	M5C	3.3 V	CMOS	1 - 16
APDL1	M5C	3.3 V	PECL	1 - 16

PECL is defined as positive- or pseudo-ECL. Table 2 summarizes the performance of the H4CPlus, H4EPlus and M5C APLL macros. The "Output Frequency Range" is the *linear* range of the VCO; its full range extends somewhat further. "Max clk tree delay" is the maximum delay that the APLL can handle in its feedback loop before going unstable.

Note 1: All of the performance numbers in Table 2 are preliminary! The guaranteed values for these parameters are in the respective H4CPlus, H4EPlus and M5C Design Reference Manuals.

Note 2: On H4CPlus and H4EPlus arrays which may use both 3.3V and 5V power, the APLL I/O must be powered by the same voltage level as the array core.

All H4CPlus, H4EPlus and M5C Series APLLs require the following six pins (see Figure 2):

- AVDD: analog power
- AVSS: analog ground
- FREF: reference frequency input pin (also used by tester to clock the core logic)
- TESTSEL: configures the APLL for tester measurements
- TESTOUT: divided-down APLL output frequency for tester
- VCOCTL: for measuring VCO control voltage and charge pump current

TESTOUT, TESTSEL, and VCOCTL are dedicated test pins which must be grounded during normal system operation. An additional input pin, FREFB, is required if the reference frequency is a PECL differential clock (see Figure 3). Each APLL also has the following five signals which interface to the array core:

- FREF_CORE: output of FREF pin input buffer; drives FREF_MUX directly, or through a PLLDELAY macro to cancel phase error due to a core divider (see Section 4.1)
- FREF_MUX: phase detector reference frequency input
- FVCO: VCO output frequency
- FVCO_DIV2: FVCO frequency divided by 2
- FFB: phase detector feedback frequency input

Referring to Figure 2 and Figure 3, for each type of APLL (CMOS-input and PECL-input), a buffer (buffer B) comparable to the FREF input buffer is included at the FFB feedback input to the phase detector in order to prevent the FREF input buffer's prop delay from adding to the phase error between the FREF pin and the clock tree. In addition, the PLLDELAY macro can be placed in the array core between the APLL's FREF_CORE output and FREF_MUX input in order to prevent a core divider's prop delay from adding to the phase error between the FREF pin and the clock tree (see Section 4.1). No external components are required for filtering of the VCO control voltage. Up to two APLLs can be used on an H4CPlus or H4EPlus array, in the lower left and upper right corners where they are isolated from digital pwr/gnd/signal interconnects to minimize coupling of digital noise into the APLL. If only one APLL is used on an array, the APLL must reside in the lower left corner.

2.1 APLL Macrocell Descriptions

The APLL macro symbol is shown in Figure 1 and is supported in Motorola's H4CPlus or H4EPlus Series library. Three test pins are utilized to bypass the APLL to test the core logic and to test the APLL. The three test pins are tied to AVSS in the application.

On M5C Series arrays, up to three APLLs can be used. If only one APLL is used, it must reside in the upper left corner. If two APLLs are used, they must reside in the upper left and upper right corners. If three APLLs are used, the only restriction is that there is no APLL in the lower left corner.

As shown in Figure 4, an APLL macro covers the corner and also four adjacent I/O sites (five I/O sites if FREF is a PECL input - see Figure 5). Accordingly, the pad locations are fixed for the APLL I/O signals. The Manufacturing Rules Verification (MARV) program contained in Motorola's OACS™ system checks that the designer has made correct pin assignments for the APLL I/O. ERC also checks compliance with the APLL placement restrictions described in the previous two paragraphs.

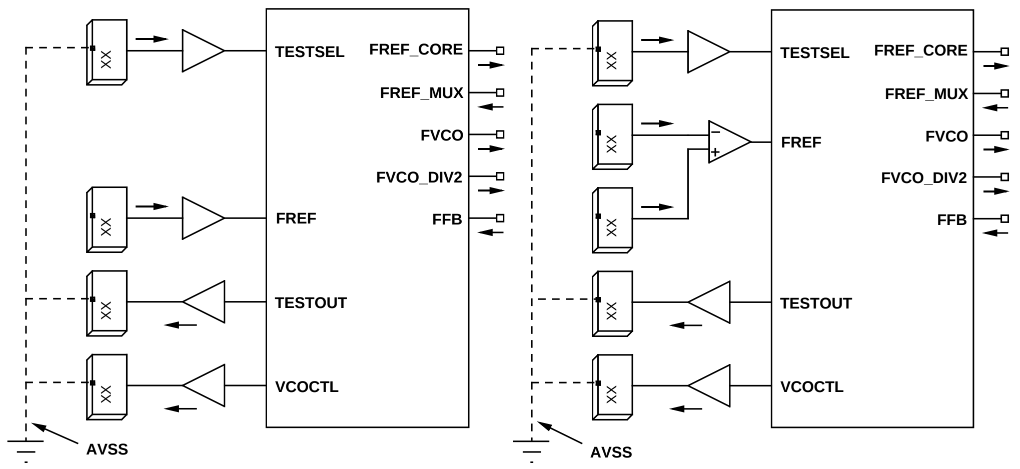


Figure 1 Analog PLL Macro Symbols

Table 2 APLL Performance*

	H4CPlus, H4EPlus		M5C
	3.3 V	5 V	
Output Frequency Range			
FVCO (MHz)	60 - 160	70 - 250	100 - 300
FVCO_DIV2 (MHz)	30 - 80	35 - 125	50 - 150
Output Duty Cycle			
FVCO	25% - 75%	25% - 75%	25% - 75%
FVCO_DIV2	50%	50%	50%
Loop Divider Value, N			
APxx1 macros	1 - 4	1 - 4	1 - 16
APxx2 macros	5 - 16	5 - 16	
Reference Frequency Range (MHz)			
Normal use:			
APxx1 macros	15 - 160	17.5 - 250	6.25 - 300
APxx2 macros	3.8 - 32	4.4 - 50	
On tester (N=8) all macros	7.5 - 20	8.75 - 31.2	12.5 - 37.5
Phase Error			
CMOS Singe-Ended Inputs	50ps	50ps	50ps
PECL Differential Inputs	200ps	200ps	200ps
Jitter	200ps	200ps	200ps
Max. Clock Tree Delay (Worst-Case)	25ns	20ns	20ns
Max. Lock-Acquisition Time	10µs	10µs	10µs

* All specs are preliminary.

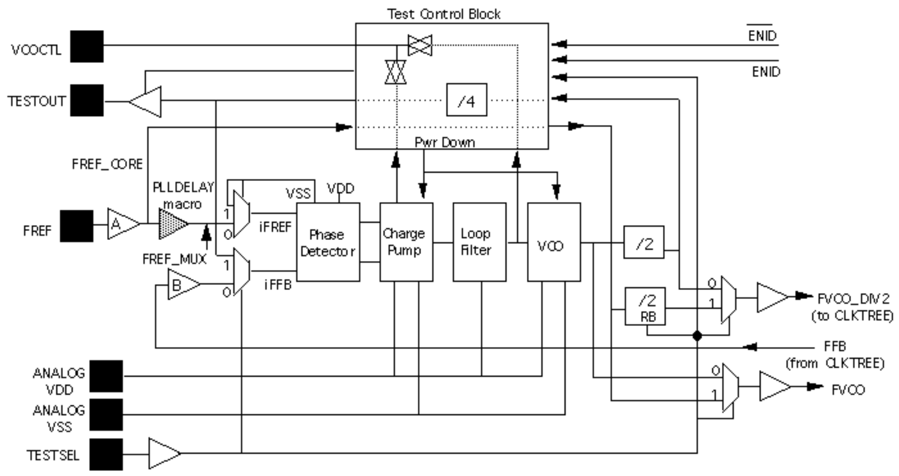


Figure 2 Analog PLL Block Diagram (CMOS Input)

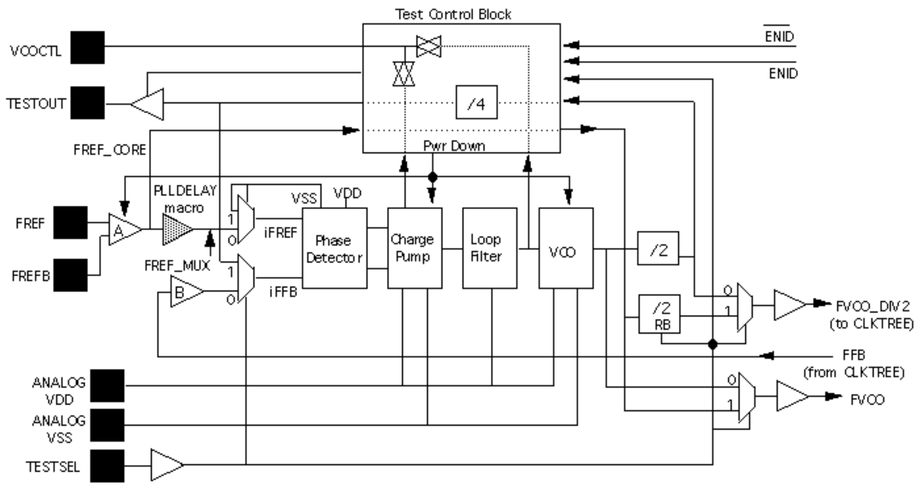


Figure 3 Analog PLL Block Diagram (PECL Input)

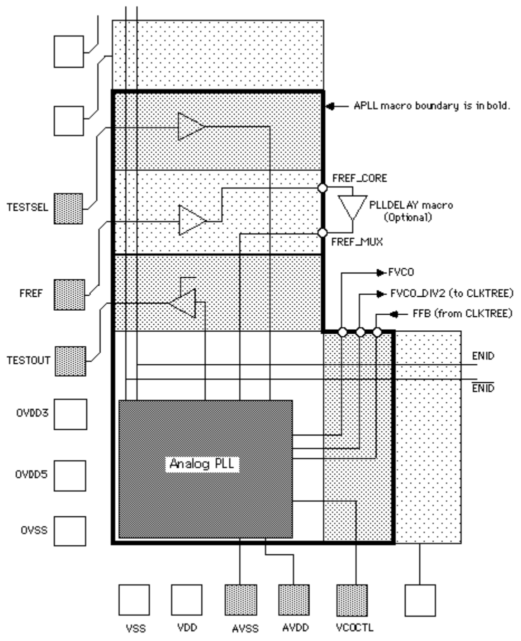


Figure 4 Analog PLL Layout (CMOS Input)

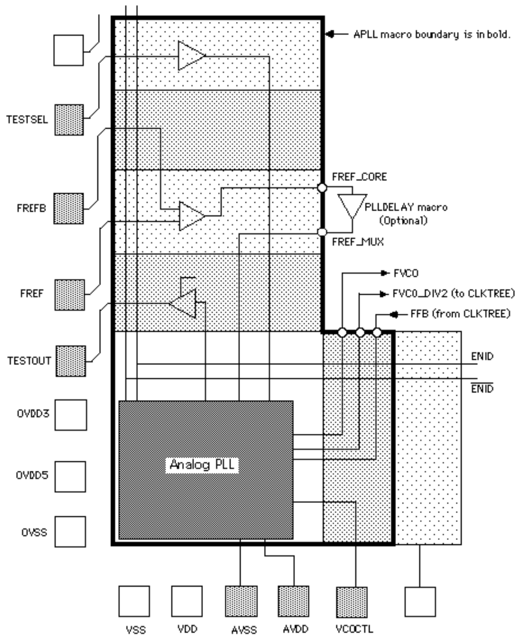


Figure 5 Analog PLL Layout (PECL Input)

3. Design Considerations

3.1 APLL Application

Figure 6 contains a typical application of the APLL. The divider blocks (÷ L and ÷ M) are used to adjust for desired clock frequencies and to center the APLL FVCO and FVCO_DIV2 outputs. The PLLDELAY macro is a delay element for matching the delay of the M divider block when M > 1. As an example, assume a 5.0 Volt core, and a 40 MHz clock tree is desired. With an input reference frequency of 20 MHz, L = 2 and M = 2.

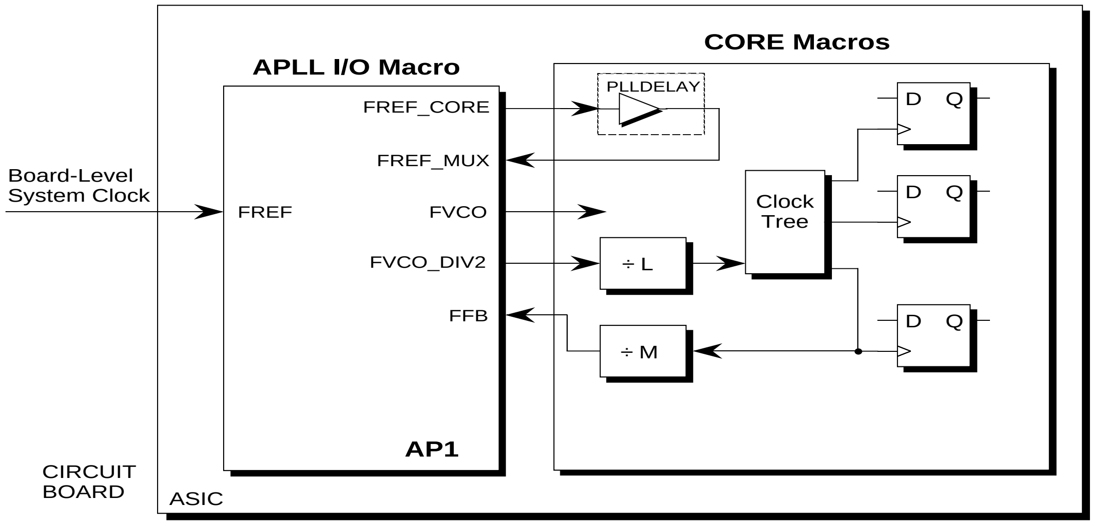


Figure 6 Clock Distribution with APLL

By selecting N = 8, (2 x 2 x 2), the FVCO_DIV2 is forced to 80 MHz. This is approximately the middle of the operation frequency range, since the FVCO_DIV2 range is between 35 and 125 MHz. (In this example, macro PLLDELAY is used to adjust delay times.)

3.1.1 APLL Operation

Figure 7 contains a block diagram of the APLL macro. Basically, the APLL is a classical second order system that compares the phase of the input reference clock (FREF) with the phase of the feedback signal (FFB), and adjusts the phase of the FFB signal to be locked in phase and frequency with the FREF signal. It uses a type IV phase/frequency detector that sends correction pulses to a charge pump. The charge pump, based on the correction pulses, either adds or subtracts charge from the on-chip passive loop filter, thereby altering the control voltage of the VCO. The VCO, in turn, produces a different phase and frequency which is fedback to the phase detector. Correction pulses are generated until the APLL is locked. Frequency multiplication is easily implemented by putting a digital divider in the feedback path.

3.1.2 APLL Power Supply

A separate analog power supply is not necessary to provide power to an APLL; however, to ensure a jitter free APLL operation, the analog AVDD and AVSS pins must be noise free. The ideal noise rejection circuitry is design/board environment dependent. The following two schemes are recommended as possible solutions.

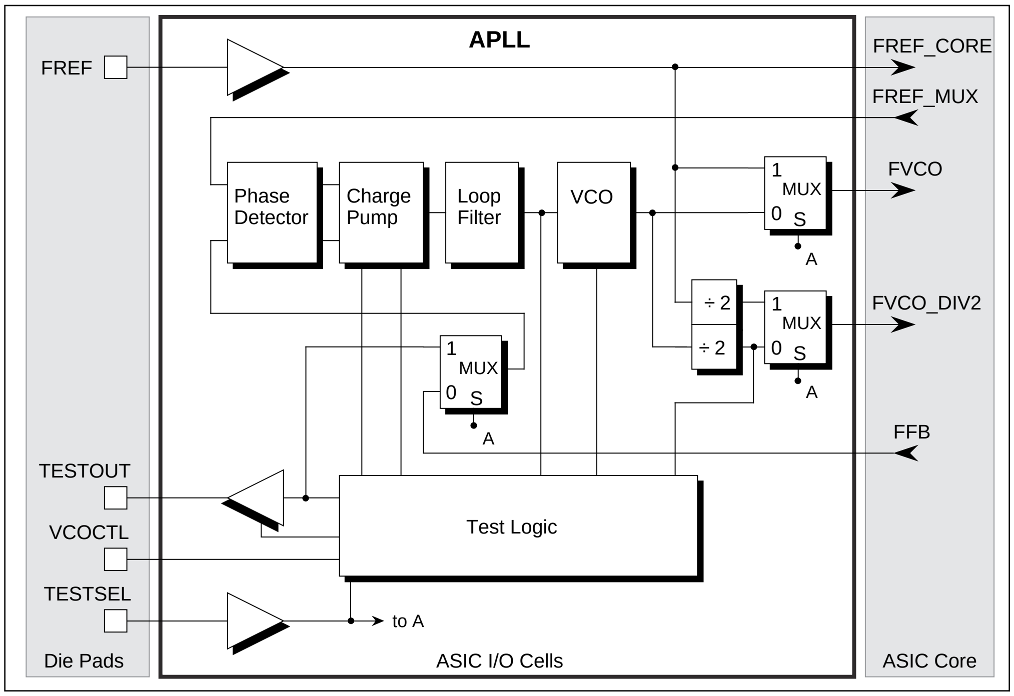


Figure 7 APLL Block Diagram

3.1.2.1 Isolation Scheme 1

Figure 8 shows an analog isolation scheme 1 which can be effective in most applications.

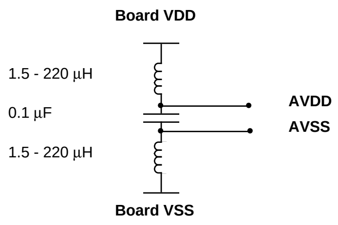


Figure 8 Analog Isolation Scheme 1

The inductors are necessary to ensure jitter-free (<600 ps) operation in a digital environment. The VCO has a gain of approximately 25 MHz/V. It is extremely sensitive to any noise on the power and ground planes. Surface mount inductors were used on production boards to successfully isolate the analog portion from a noisy digital environment with long-term jitter being less than ±300 ps.

A range of inductor values (1.5 -- 220 µH) was specified since it is impossible to predict the magnitude and frequency of noise present in every system. Surface mount inductors with identical footprints are available in this range of values from several vendors. Several inductor manufacturers and respective part numbers are listed in Table 3

In addition to the 0.1 µF bypass capacitor shown in the analog isolation diagram, there should be a 0.1 µF bypass capacitor between each of the other (digital) four V_{CC} pins and the board ground plane. This will reduce output switching noise caused by the 88915 outputs, in addition to reducing potential for noise in the "analog" section of the chip. These bypass capacitors should also be tied as close to the 88915 package as possible.

Table 3 Inductor Manufacturers, Part Numbers, and Selected Specifications

Inductor Manufacturer	Part Number	Inductance	Self Resonant Frequency	Footprint (W x L x H)	Telephone Number
Coilcraft Coilcraft	1812CS-822 1812LS-224	8.2 µH 220 µH	80 MHz 6 MHz	0.15 x 0.195 x 0.135 (in) 0.15 x 0.195 x 0.135 (in)	(708) 639-6400
Dale Dale	IMC-1812 1.5 H ± 10% IMC-1812 220 H ± 10%	1.5 µH 220 µH	70 MHz 4 MHz	0.126 x 0.177 x 0.126 (in) 0.126 x 0.177 x 0.126 (in)	(605) 665-9301
Toko Toko	380LB-1R5K 380HB-221K	1.5 µH 220 µH	75 MHz 3.9MHz	2.5 x 3.2 x 2.2 (mm) 2.5 x 3.2 x 2.2 (mm)	(708) 297-0070
Murata-Erie Murata-Erie	LQH3C2RM03M00-01 LQH3C221K03M00-01	2.2 µH 220 µH	64 MHz 6.8 MHz	2.5 x 3.2 x 2.0 (mm) 2.5 x 3.2 x 2.0 (mm)	(404) 436-1300

3.1.2.2 Noise Filter Scheme 2

Figure 9 shows a noise filter scheme 2 associated with the analog VDD port.

For both schemes suggested, the components involved should be tied as close to the associated analog pin(s) as possible.

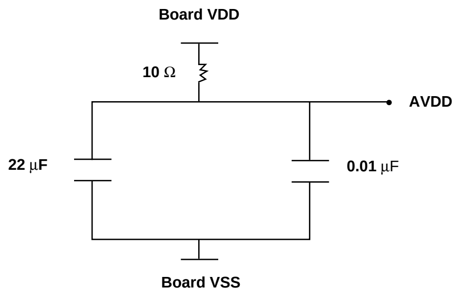


Figure 9 Noise Filter Scheme 2

4. APLL Modelling for Simulation

4.1 Overview

Figure 10 is a generic block diagram showing clock distribution using an APLL. Either or both of the divide-by-L and divide-by-M may be used. If used, they reside in the array core. The phase detector reference frequency iFREF is actually an internal signal in the APLL. As shown in Figure 2 and Figure 3, iFREF drives directly into the phase detector and is delayed from the APLL's FREF_MUX input port by a mux prop delay. Similarly, iFFB is actually an APLL internal signal which connects directly to the phase detector and is delayed from the APLL's FFB input port by the prop delay through a buffer and a mux. These mux and buffer delays are such that when the APLL has phase-locked iFFB to iFREF, then the FREF pin will be phase-locked to the clock tree output, which is the ultimate objective. A special PLLDELAY macro can be used to cancel phase error between the clock tree and FREF which is caused by the divide-by-M. The PLLDELAY macro has the same delay as the CK->Q of a resettable flip-flop, therefore if a divide-by-M is used it should be designed using resettable flip-flops.

Note: Use of another divider in place of the PLLDELAY macro is not supported. The Motorola-internal vectors used to test the APLL in silicon require that the frequency at iFREF be the same as the frequency at the FREF pin. See Section 4 for details of the test strategy for APLL arrays.

The feedback loop between the VCO and phase detector resides in the array core, external to the APLL, and contains the clock tree and possibly a frequency divider, which will be referred to as the core divider. If a core divider exists it typically would follow the clock tree as does the divide-by-M. However the core divider could also precede the clock tree, as does the divide-by-L, if the clock tree is to be driven by a frequency lower than the minimum possible FVCO_DIV2 from the APLL. A third possibility is that the core divider is composed of both the divide-by-L and divide-by-M.

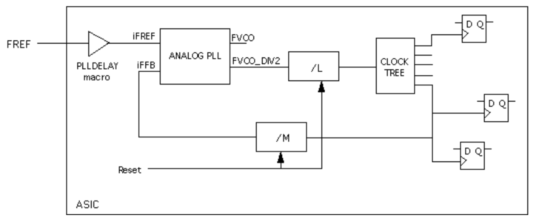


Figure 10 Clock Distribution Using an Analog PLL

In addition to generating the VCO frequency FVCO, the APLL contains a divide-by-2 to generate FVCO_DIV2, which has a 50% duty cycle. FVCO_DIV2 typically is the signal used to drive the clock tree, where FVCO is available for faster clocking of a small, localized block of logic. Therefore, throughout this document it is assumed that the clock tree is driven by FVCO_DIV2 rather than FVCO. In this case, FVCO gets divided by 2 (within the APLL itself) and then divided again by the core divider, if one exists, before arriving at the phase detector feedback input FFB. The product of these two divider values equals the *loop divider* value “N.” The APLL model measures the reference frequency iFREF and the loop divider value N and generates VCO frequency required for phase-lock, FVCO = N x iFREF. The phase of FVCO_DIV2 compensates for the clock tree plus core divider delay in the core feedback loop such that the output of the clock tree is in phase with the board reference clock at the FREF pin.

In its default mode, the model acquires phase-lock approximately 20 cycles after the start of iFREF (or after reset of the core divider eliminates its ‘X’ state at simulation startup). However, if the user prefers, the model can also be set-up to emulate the actual time required by the APLL to achieve phase-lock in the real-world. During this “acquisition delay” the model puts out a constant (but not phase-locked) VCO frequency, which will change abruptly to the phase-locked frequency FVCO = N x iFREF after 10us has expired. Other than accurate acquisition delay, this behavior does not model the true transient response of the APLL. However, what is important is accurate modeling of the APLL’s steady-state performance after phase-lock has been achieved.

The model generates FVCO and FVCO_DIV2 such that after phase-lock is achieved the clock signal fed back to the phase detector, iFFB, has the specified worst-case phase error relative to the phase detector reference clock, iFREF. The user can select this steady-state phase error to be leading, lagging, or randomly jittering between the two as described in Section 4.6. The model also does a variety of checks for such things as loss of phase-lock, the FVCO frequency required is out of range, etc.

The Verilog model emulates the APLL only during system simulations and not during option release simulations, which generate test vectors used for testing of parts. The reason is that the APLL is inactive during tester application of option release test vectors, which verify all circuitry except for the APLL. Consequently, during option release simulations the clock applied at the APLL’s FREF pin will bypass the APLL and drive the core directly. For information on how to control the APLL during option release simulations, as well as information on how the APLL is verified on the tester, see Section 4, “Test Strategy for APLL Arrays.”

4.2 Initialization/Reset of Dividers

When an APLL array is on a board in a system, it is unnecessary to reset the two dividers in Figure 10. However, during *system-mode simulation* these dividers must be initialized to a known state before the FREF and FVCO clocks can propagate to phase detector inputs iFREF and iFFB, respectively. Unfortunately, prior to phase-lock, FVCO and FVCO_DIV2 have no fixed timing relationship with respect to the chip’s input pins. Consequently, trying to do a synchronous hardware initialization/reset of the core divider may be difficult to do without generating timing violations, such as a reset recovery time violation. A more practical approach during system-mode simulations *(but not option release simulations)* would be to use the Verilog “force” and “release” commands to initialize the states of the flip-flops in the core divider. This can be done by “forcing” the D inputs of the divider flops to known states until FVCO starts, at which time these states will get clocked into the flops. When “release” occurs the flops are released to function normally. “Release” can occur at any time with respect to the arrival of clock edges at the core divider without causing the divider state to go unknown.

Alternatively, an asynchronous set/reset of the dividers can be done via chip logic or a pin at simulation start-up, *before the iFREF clock starts toggling*, since the model will not generate an FVCO clock until iFREF starts to toggle. In this way an asynchronous set or reset of all dividers can be done without generating timing violations.

Note: The reset signal for these dividers cannot be shared with any circuitry that must be reset after phase-lock is acquired, since resetting the APLL’s dividers would cause the APLL to lose phase-lock.

Artificial initialization of the core dividers using “force” and “release” can be used for system-mode simulations *but not for option release simulations*, where simulation output states must match chip output states on the tester. By driving the clock tree, the divide-by-L in Figure 10 affects chip output states. Therefore *during option release simulations* the divide-by-L must be initialized/reset via chip logic or a pin, and not by using “force” and “release.” The same is true of the divide-by-M if it is made observable at an output pin in order to test it. If the divide-by-M drives only FFB then it affects no output pin during option release simulations and is therefore not testable (since the APLL is inactive). In this case it need not be initialized. The divide-by-M still needs to be initialized/reset during system-mode simulations, however.

4.3 Acquisition Mode

The APLL model starts in acquisition mode at simulation start-up. It measures the frequency of the phase detector reference clock, iFREF, as well as the loop divide-by-N in order to calculate the required VCO lock frequency FVCO = iFREF x N. The model starts generating an FVCO clock which has an arbitrary phase relationship to iFREF. The resulting feedback clock at the phase detector, iFFB, has an initial phase error with respect to iFREF. The model measures this phase error and corrects the phase of FVCO such that iFFB will be in phase with iFREF, producing phase-lock.

At the start of simulation the model waits for a clock signal to appear at iFREF, and then measures the period of iFREF by keeping track of the time between successive iFREF rising edges. The model now starts generating FVCO and FVCO_DIV2, where FVCO is the center frequency of the VCO. While the VCO free-runs, the model waits until the state at iFFB is no longer 'X,' indicating that the core divider has been initialized to a known state as described previously. The model then waits until a 0->1 rising edge occurs at iFFB (as opposed to an X->1 rising edge), indicating that the core divider has been released to function normally after having been initialized/reset.

When the second iFFB rising edge occurs the model measures the frequency at iFFB and calculates the loop divider ratio 'N', where N = FVCO frequency/(iFFB frequency). If N is not within the specified range for the APLL macro used (see Section 2), the model stops the simulation after printing a message to the effect that the user must modify the loop divider circuitry such that N does lie within the specified range. If N is within the specified range but iFREF x N = FVCO is not within the specified frequency range for the VCO, the user must modify FREF and/or N such that FVCO does lie within the VCO's range. To modify N, circuitry in the array core must be changed; however, FREF can be modified interactively during Verilog simulation as described in Section 4.6. If iFREF x N = FVCO is, in fact, out of the VCO's range the model will now return to the start of the acquisition mode. Otherwise operation proceeds as follows.

Once a "legal" loop divider ratio N has been determined, the following information is printed to the screen:

```
- Loop divider value, N
- Phase detector reference frequency, iFREF
- VCO frequency, FVCO
- VCO/2 frequency, FVCO_DIV2
- Duration of FVCO high and low pulses (FVCO duty
  cycle, effectively)
```

Then a series of pulses is generated at FVCO for use in measuring the propagation delay through the feedback loop, which is equal to the sum of the clock tree propagation delay and loop divider propagation delay.

This is done in order to verify that the loop delay is not so large as to cause the APLL to go unstable and never acquire phase-lock. After generating an FVCO pulse the model waits long enough to see if the FVCO pulse causes a rising edge at iFFB. It will take anywhere from 1 to N FVCO pulses to generate a rising edge at iFFB, depending on the initial state of the loop divider. When a rising edge does occur at iFFB, the loop delay is measured as the time delay between the rising edge at iFFB and the last FVCO rising edge. If the loop delay is larger than the specified limit, the model prints a message to that effect and stops the simulation to allow the clock tree or loop divider to be re-designed. Otherwise the model will now begin its 10us acquisition delay, as described in Section 4.1. At the end of this delay, the VCO stops long enough for the clock tree to empty of all pulses generated by the free-running VCO during the acquisition delay. (If the APLL was not set-up to emulate the real-world acquisition delay, the model will skip down to this point if the feedback loop delay measured was within spec.) The APLL model now waits for the next rising edge of iFREF to start generating N cycles of the VCO lock frequency FVCO = iFREF x N. N cycles of FVCO span a complete cycle of iFREF, and the last of these N FVCO cycles should produce the next rising edge at iFFB (due to the state in which the loop divider was left after the loop delay was measured). The rising edge of the first of these N FVCO cycles is delayed from the iFREF rising edge by the "VCO_offset" such that the resultant iFFB rising edge is aligned with a subsequent iFREF rising edge (within the APLL's specified phase error), producing phase-lock. The model calculates the VCO_offset using the previously measured feedback loop delay.

If the Nth FVCO pulse does not produce a rising edge at iFFB, something's probably wrong with the core divider; for example, it may have been disabled or reset. In this case, the APLL will print an error message to that effect and then restart its acquisition routine to try again to acquire phase-lock. If, on the other hand, phase-lock has indeed been acquired, the following information is printed to the screen:

```
- Time at which phase-lock was acquired.
- APLL steady state phase error at iFFB with
  respect to iFREF.
```

Now the APLL model goes into tracking mode.

4.4 Tracking Mode

Whenever a rising edge occurs on iFREF, the model measures the time difference between this edge and the associated rising edge on iFFB. If this "phase error" is less than the specified worst-case phase error of the APLL, then the APLL is still in lock. In this case the model will generate the next N FVCO cycles in the manner described previously in Section 4.3, in order to produce the next rising edge on iFFB. However if the phase error between iFREF and iFFB is greater than the specified worst-case phase error of the APLL, lock has been lost. In this case the model prints a "loss-of-lock" message which includes the simulation time at which lock was lost. The model waits long enough for the clock tree to empty of all 'pipelined' FVCO pulses, and then returns to acquisition mode to try to re-acquire phase-lock.

4.5 Initialization of APLL Simulation Parameters

For best accuracy, Verilog simulations involving APLL's in system mode should be done with the following timescale setting: `'timescale 1ns/1ps`

Therefore the timescale statement in the *asic_verilog* 'verilog.control' file should be changed to 1ps resolution, as shown above. For option-release simulations, the timescale can be left at the default value of 10ps.

In addition, there are four user-settable parameters whose range of values are hard-coded inside the APLL Verilog model because they cannot be specified in the Standard Delay Format (SDF) *verilog.timing* file output by DECAL. These parameters are:

- ***jitter*** -- determines whether iFFB will always lead, always lag, or randomly jitter between leading and lagging with respect to iFREF. The amount of lead or lag is always equal to the APLL's maximum steady-state phase error. Valid values for *jitter* are "lead", "lag" or "random". The default value is "random".
- ***use_silicon_delay*** -- determines whether the model will emulate the real-world APLL acquisition delay (described in Section 4.1). Valid values for *use_silicon_delay* are "yes" or "no". The default value is "no".

- ***vco_duty_cycle*** -- determines the duty-cycle of the FVCO output for this simulation. Valid values for *vco_duty_cycle* are “min”, “typ” or “max”. The default value is “min”.
- ***ptv*** -- determines whether the best-, typical-, or worst-case **p**rocess/**t**emperature/**v**oltage (PTV) value is to be used for the *maximum feedback loop delay*. Valid values for *ptv* are “bst”, “typ” or “wst”. The default value is “wst”.

These four parameters are used only during system-mode simulations. They are not used during option release simulations, during which the FREF input clock bypasses the APLL and drives the clock tree directly. (See Section 4 for details regarding option release simulations.)

In a non-interactive simulation, the value used for *maximum feedback loop delay* is determined by one of the following Verilog command line “plus arguments”: +mindelays, +typdelays or +maxdelays. Therefore if the OACS tool *asic_verilog* is used, the *maximum feedback loop delay* value will be chosen automatically according to the PTV conditions selected for the array; the *ptv* parameter is ignored. However in an interactive simulation, the *maximum feedback loop delay* value is determined by assigning the *ptv* parameter a value of “bst”, “typ” or “wst”. In this case the designer must set the *ptv* parameter value to match the PTV conditions chosen for the array. Otherwise the value for *maximum feedback loop delay* may be for a different PTV condition than that used for the rest of the array.

The FVCO_DIV2 output has a 50% duty cycle, but the FVCO duty cycle can vary over a wide range. Designs which use the APLL’s FVCO output, system-mode simulations should be done at the following four sets of conditions:

i) PTV = best-case, `vco_duty_cycle = “min”`

ii) PTV = best-case, `vco_duty_cycle = “max”`

iii) PTV = worst-case, `vco_duty_cycle = “min”`

iv) PTV = worst-case, `vco_duty_cycle = “max”`

Because of the way that the APLL model generates “random” jitter, it is possible that the model will falsely swallow low-going FVCO pulses when *vco_duty_cycle* = “max”, if the FVCO is operating at the upper end of its frequency range. These two conditions, coupled with random jitter, can combine to make the low-going FVCO pulses narrow enough that they get swallowed by the FVCO output buffer within the APLL model. In such cases, if FVCO is used in the design then the *jitter* parameter must be restricted to values of “lead” or “lag”.

The user can assign a value to a particular parameter by putting a ‘defparam’ statement in the HDL stimulus file, such as:

```
defparam stim.cell1.\TC_TOP/APLL.448P_4
   .core_apll.ptv = “wst”;
```

The pathname is taken from a real design named “TC_TOP.” “stim” is the name of the module which applies stimulus to “TC_TOP”, “cell1” is the name of the instantiation of “TC_TOP” within module “stim”, “\TC_TOP/APLL.448P_4” is the instance name generated by the OACS NETLIST tool for the APLL macro used in “TC_TOP”, and *core_apll* is the sub-module within the APLL Verilog model in which the *ptv, jitter, vco_duty_cycle,* and *use_silicon_delay* parameters are defined. Note that in this particular design a space is required after the APLL instance name because its first character is a backslash. Similar statements can be used to assign values to the *jitter*, *vco_duty_cycle* and *use_silicon_delay* parameters, for example:

```
defparam stim.cell1.\TC_TOP/APLL.448P_4
   .core_apll.jitter = “lead”;
defparam stim.cell1.\TC_TOP/APLL.448P_4
   .core apll.vco_duty_cycle=“max”;
defparam stim.cell1.\TC_TOP/APLL.448P_4
   .core apll.use_silicon_delay=“yes”;
```

Alternatively, these four parameters can be changed “on the fly” within an interactive Verilog run if the designer wishes to re-simulate without having to re-compile. If such a re-simulation is to be at a different PTV, the *ptv* parameter must be changed accordingly. The following interactive Verilog commands show how to change these parameters prior to a re-simulation (“>” represents the Verilog prompt in interactive mode):

```
> $reset;
> $scope(stim.cell1.\TC_TOP/APLL.448P_4
   .core_apll);
> ptv = “wst”;
> jitter = “lead”;
> vco_duty_cycle = “max”;
> use_silicon_delay = “yes”;
> .
```

The designer may even want to change one of these three parameters prior to the first simulation after compilation. If so, a *$stop* command could be included at the start of the HDL stimulus to cause Verilog to stop at time zero and give a “>” prompt. At this time the designer can enter the same interactive Verilog commands shown above, although in this case the *$reset* command is unnecessary. Alternatively, ‘defparam’ statements can be put in the HDL stimulus file, as described previously.

This same method can be used to change FREF interactively if necessary. If the VCO frequency FVCO = iFREF x N is out of the APLL’s range, the APLL model prints a message to the screen stating that iFREF and/or N must be changed. Referring to Figure 10, changing N
(N = L x M) requires a circuit change. However if the designer chooses to change only FREF, he can do so by scoping into his HDL stimulus module and updating the FREF period parameter.

4.6 Example Simulations of an APLL

Figure 11 shows the Verilog graphical waveforms for an interactive system mode simulation using an APLL. Also shown is a portion of the transcript window containing messages printed out by the APLL model. For this simulation, the

APLL model has been set-up to not emulate the real-world acquisition delay of the APLL. Note that the TSTSEL waveform, which corresponds to the TESTSEL input on the APLL, is held low throughout the simulation. (For an option release simulation, TSTSEL = TESTSEL would be taken high after the first test cycle, and held high throughout the rest of the simulation. See Section 5.1, “Testing the Array Core”.) Referring to Figure 10, at simulation start-up the APLL is configured with L = 1 and M = 2.

In this example, the divide-by-M is reset by a RESETB signal (top waveform in Figure 11) rather than by the “force” and “release” commands, which were previously described in Section 4.6. While the divide-by-M is held in reset iFFB remains low.

Initially, FVCO and FVCO_DIV2 are unknown, as is FFB prior to reset of the loop divider. The loop divider consists of a divide-by-2 flip-flop in the array core along with the flip-flop internal to the APLL which divides FVCO down to FVCO_DIV2. When RESETB goes active the loop divider state becomes known, at which time the APLL model outputs a low on both FVCO and FVCO_DIV2. Now the loop divider’s reset signal, RESETB, can return to its inactive state, since there is no longer an ‘x’ at the clock input to the loop divider. Since FVCO and FVCO_DIV2 will start toggling as soon as FREF starts toggling, FREF is not started until after RESETB goes inactive in order to prevent reset recovery time violations in the loop divider (discussed in Section 4.2).

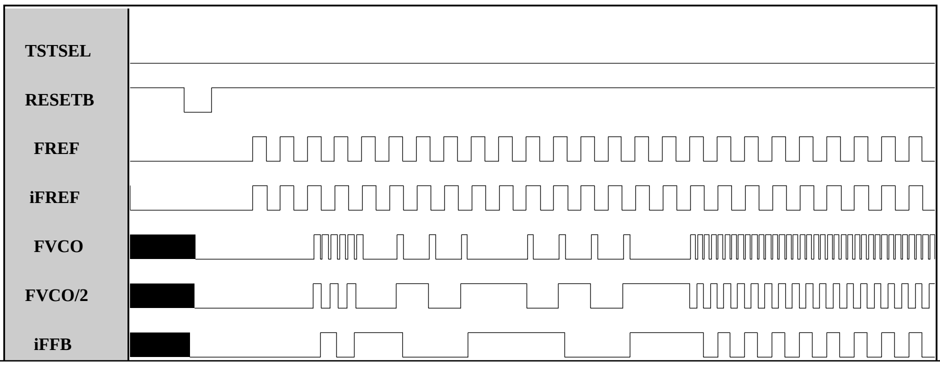
The APLL model waits for the first two iFREF pulses to determine the frequency of iFREF and to start generating FVCO_mid and FVCO_mid/2, where FVCO_mid is the VCO center frequency. The model waits for feedback pulses at iFFB. After the second iFFB rising edge, the model measures the iFFB frequency and calculates the loop divider value.

N = FVCO_max/(iFFB frequency). N is then used to calculate the in-lock VCO frequency:

FVCO = iFREF x N. N, iFREF, FVCO, and FVCO_DIV2 are printed to the screen, as well as the high and low pulse widths of FVCO (FVCO duty cycle, effectively). At this point the model stops generating FVCO long enough for the clock tree to empty of all pulses. After this pause the model starts generating individual FVCO pulses and looking for a rising edge to result at iFFB. In this example it takes 3 FVCO pulses to cause the next rising edge on iFFB, due to the initial state of the loop divider. These three pulses are followed by N additional FVCO pulses (N happens to be four in this case) to verify that N more FVCO pulses will cause another iFFB rising edge at the expected time. At this point the model starts generating the in-lock frequency FVC0 = N x iFREF, with the proper phase such that the next rising edge of iFFB will be phase-locked to iFREF.

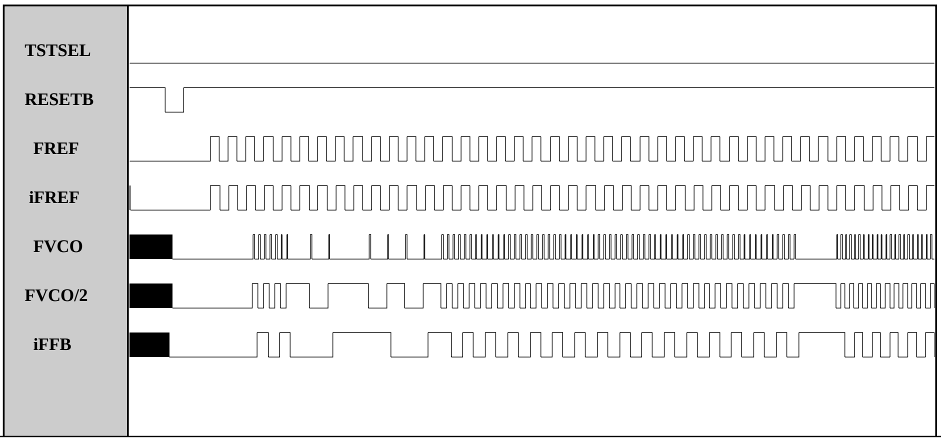
Figure 7 shows Verilog graphical waveforms and part of the transcript window for a non-interactive system mode simulation in which the APLL model has been set-up to emulate the real-world acquisition delay of the APLL. (Section 4.5 explains how to do this.) For this example simulation, the acquisition delay was shortened in order to fit the waveforms on the page. At the end of the acquisition delay the model stops generating the VCO center frequency at FVCO and waits long enough for the clock tree to empty of all pulses. After this pause the model starts generating the in-lock frequency FVC0 = N x iFREF, with the proper phase such that the next rising edge of iFFB will be phase-locked to iFREF.

Figure 8 shows Verilog graphical waveforms and part of the transcript window for the start of an option release simulation using an APLL. Note that the TSTSEL waveform, which corresponds to the TESTSEL input on the APLL, is taken high after the first test cycle and held high throughout the rest of the simulation. The FREF clock is low at simulation start-up, and stays low until after TESTSEL goes high. Similarly RESETB, which resets the loop divider, is inactive (high) until after TESTSEL goes high.



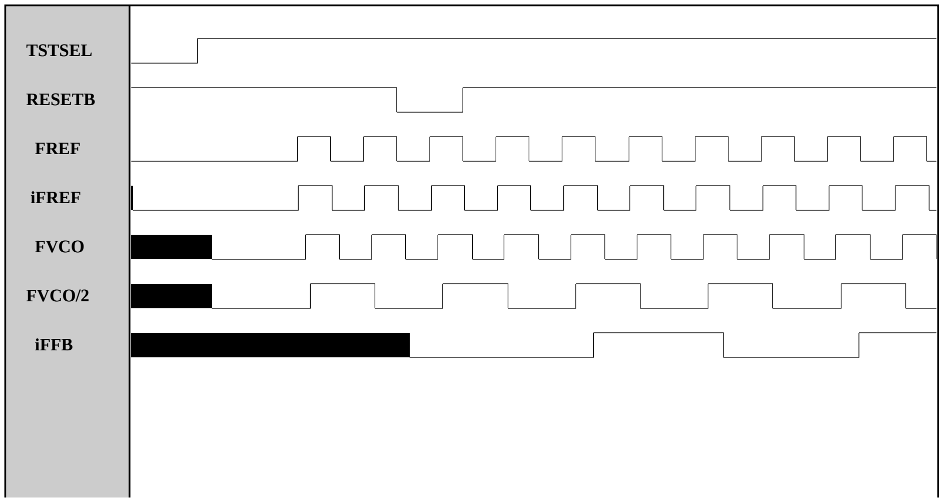
```
############################################################################
############################################################################
###                                                                      ###
###  Parameter ‘ptv’ has been given a value of “wst” for APLL instance  ###
###      stim.cell1.APLLOACS22/APLL.27P_1.core_apll
###  Worst-case timing must also be used for the rest of the design.    ###
###                                                                      ###
############################################################################
############################################################################

Assigned values for user-settable parameters for APLL instance
     (stim.cell1.APLLOACS22/APLL.27P_1.core_apll):
   vco_duty_cycle = “max”
   jitter = “lag”
   use_silicon_delay = “no”

############################################################################
############################################################################

   To backannotate timing before simulation -
   type one of the following :

    (If more than one is activated, last choice
    overrides the rest.)

    Type “ba_timing_wcs;” for worst case timing
    Type “ba_timing_typ;” for typical case timing
    Type “ba_timing_bcs;” for best case timing


   Else, type “.” (period) to simulate with
   unit timing.
L84 “aplloacs22/design_data/verilog.control”: $stop at simulation time 0.00 ns
Type ? for help
C1 > ba_timing_wcs;
C2 > .

    Back-annotating worst case timing .................
              .
              .
              .
Operating environment for APLL instance
        (stim.cell1.APLLOACS22/APLL.27P_1.core_apll.acquire):
   N = 4 (Loop divider value)
   iFREF  = 50 MHz (Phase detector reference frequency)
   FVCO   = 200 MHz (VCO frequency)
   FVCO_DIV2 = 100 MHz (VCO/2 frequency)
   VCO_high = 3.75 ns (VCO high pulse width)
   VCO_low  = 1.25 ns (VCO low pulse width)

APLL instance (stim.cell1.APLLOACS22/APLL.27P_1.core_apll.track)
    ACQUIRED PHASE-LOCK at time 447.46 with phase error = 0.25 ns

L97 “/home/cdc1/aplltc/tc_top/sub_blocks/aplloacs22/vectors/stimulus/hdl.stim”:
$stop at simulation time 590.00 ns
C2 >
```

Figure 11 Example Verilog System-Mode Simulation of an APLL (acquisition delay excluded)



```
##########################################################################
##########################################################################

Assigned values for user-settable parameters for APLL instance
     (stim.cell1.APLLOACS22/APLL.27P_1.core_apll):
   vco_duty_cycle = “min”
   jitter = “lead”
   use_silicon_delay = “yes”

##########################################################################
##########################################################################

    Back-annotating worst case timing .................


   *** SDF Annotator version 1.0.10
   ***    SDF file:  aplloacs22/timing/veritool.timing
   ***    Back-annotation scope:  stim
   ***    Configuration file:  /home/bass1/oacs2.x/oacs_apll_tools2.0/sdf.config
   ***    SDF Annotator log file:  aplloacs22/timing/reports/sdf.log
   ***    MTM selection parameter specified: MAXIMUM

   ***    SCALE FACTORS parameter specified: 1.000000:1.000000:1.000000

   ***    SCALE TYPE parameter specified: FROM_MTM

          Parsing configuration file...

          Configuring for back-annotation...

          Reading SDF file and back-annotating timing data...

   *** SDF back-annotation successfully completed

Operating environment for APLL instance
        (stim.cell1.APLLOACS22/APLL.27P_1.core_apll.acquire):
   N = 4 (Loop divider value)
   iFREF  = 50 MHz (Phase detector reference frequency)
   FVCO   = 200 MHz (VCO frequency)
   FVCO_DIV2 = 100 MHz (VCO/2 frequency)
   VCO_high = 1.25 ns (VCO high pulse width)
   VCO_low  = 3.75 ns (VCO low pulse width)

APLL instance (stim.cell1.APLLOACS22/APLL.27P_1.core_apll.track)
    ACQUIRED PHASE-LOCK at time 824.46 with phase error = -0.25 ns

L128 “/home/cdc1/aplltc/tc_top/sub_blocks/aplloacs22/vectors/stimulus/hdl.stim”:
 $stop at simulation time 900.00 ns
Type ? for help
C1 >
```

Figure 12 Example Verilog System-Mode Simulation of an APLL (acquisition delay included)



```
##########################################################################
##########################################################################

Assigned values for user-settable parameters for APLL instance
     (stim.cell1.APLLOACS22/APLL.27P_1.core_apll):
   vco_duty_cycle = “min”
   jitter = “random”
   use_silicon_delay = “no”

##########################################################################
##########################################################################


    Back-annotating worst case timing .................


   *** SDF Annotator version 1.0.10
   ***    SDF file:  aplloacs22/timing/veritool.timing
   ***    Back-annotation scope:  stim
   ***    Configuration file:  /home/bass1/oacs2.x/oacs_apll_tools2.0/sdf.config
   ***    SDF Annotator log file:  aplloacs22/timing/reports/sdf.log
   ***    MTM selection parameter specified: MAXIMUM

   ***    SCALE FACTORS parameter specified: 1.000000:1.000000:1.000000

   ***    SCALE TYPE parameter specified: FROM_MTM

          Parsing configuration file...

          Configuring for back-annotation...

          Reading SDF file and back-annotating timing data...


   *** SDF back-annotation successfully completed

TESTSEL pin went high at time             24.30 ns
     for APLL instance ( stim.cell1.APLLOACS22/APLL.27P_1.core_apll )
     FREF will now bypass the APLL and appear at the FVCO output.
     (If APLL output ports FVCO_DIV2 and FVCO stay ‘x’, then
     TESTSEL was not held low long enough, unless port FREF_CORE is also ‘x’).

L96 “/home/cdc1/aplltc/tc_top/sub_blocks/aplloacs22/vectors/stimulus/hdl.stim”:
$stop at simulation time 250.00 ns
Type ? for help
C1 >
```

Figure 13 Example Verilog Option Release Simulation of an APLL

5. Test Strategy for APLL Arrays

5.1 Testing the Array Core

On the tester, customer "option release" test vectors are used to test all of the array except the APLL, which is powered down during this time. Consequently the APLL is not used to clock the array core. The ASIC designer must use an external test clock to generate test vectors for option release, as is done for any non-APLL design. This external test clock is applied at the FREF pin.

At simulation start-up TESTSEL must be low, and must stay low for at least one test cycle, in order to initialize some flip-flops inside the APLL. During this time the model must output an 'x' on APLL outputs FVCO and FVCO_DIV2, since the VCO will be oscillating freely in silicon. To ensure that FVCO and FVCO_DIV2 remain 'x' while TESTSEL is low, the following two conditions must be met:

1. FREF should be low at simulation start-up, and should remain low at least until TESTSEL goes high.
2. If there is a reset signal for the loop divider, this reset must be inactive at simulation start-up, and must remain inactive at least until TESTSEL goes high. This condition is required because as soon as a known logic state appears at iFFB, the model outputs a known (low) state at FVCO and FVCO_DIV2 to facilitate reset of the loop divider during system simulations.

After TESTSEL goes high the APLL will be powered down, and the reference clock at the FREF pin will bypass the APLL and come out at the APLL's FVCO port (see Figure 2 and Figure 3). Similarly, FREF/2 comes out on the APLL's FVCO_DIV2 port. **Once TESTSEL goes high it must stay high throughout the rest of the option release simulation.**

5.2 Testing the APLL

The APLL is tested at Motorola by a canned test routine, during which the rest of the array does not toggle. This procedure is used to eliminate coupling of digital switching noise into the APLL through AVDD and AVSS, which are tied to the core VDD and VSS on the tester in order to eliminate the need for special test hardware for APLL arrays. In a customer's system, of course, AVDD and AVSS provide isolated power and ground for the APLL.

The APLL contains a divide-by-4 which is driven by VCO/2 in order to produce a frequency at the TESTOUT pin which is slow enough to be measured on a production tester. As shown in Table A.1 in the Appendix, the following tests are performed on the tester while in APLL test mode (TSTQ1 = 1):

i) Allow the APLL to lock at its center frequency and measure the VCO/8 frequency at the TESTOUT pin, with the VCOCTL pin turned off.

ii) Allow the APLL to lock at its center frequency and measure the VCO/8 frequency at the TESTOUT pin, and the VCO control voltage at the VCOCTL pin.

iii) Allow the APLL to lock at its center frequency and measure the VCO/8 frequency at the TESTOUT pin, and the charge pump current at the VCOCTL pin.

iv) Measure the dynamic IDD of the APLL. (A CMOS input APLL still will be in phase-lock from the previous step. For a PECL input APLL, the PECL input will be turned off by ENID ("Enable IDD" pin, see OACS User/Reference Guide); therefore the dynamic IDD measurement will be made while the APLL is not phase-locked but is free running at the minimum possible VCO frequency, since the phase detector reference frequency input, iFREF, will not be toggling).

On the tester, frequency measurement is effectively done by locating an edge on TESTOUT and then examining several more cycles to see that subsequent edges occur within the expected window. CMOS-input APLL's will remain phase-locked when moving from test (i) to test (ii), and from test (ii) to test (iii), etc. However PECL-input APLL's will lose lock when moving from one test to another. As shown in Table A.1, toggling the ENID pin is what causes the transition from one test to the next. However taking ENID high also powers-down the PECL input buffer, at which time a PECL-input APLL will lose phase-lock. Therefore after ENID is taken back low to begin the next test, a PECL-input APLL must be given time to re-acquire phase-lock before measurements are taken.

Tests i-iv are repeated at the APLL minimum and maximum *operating* frequencies, which are the most extreme frequencies achievable within the linear range of the VCO transfer function. These frequency limits are given in Section 2.

The VCOCTL pin is used to measure the VCO control voltage and charge pump current. These measurements can be related to the stability and bandwidth of the APLL. This pin should be tied to analog VSS in the customer's system to prevent noise from being injected onto the VCO control voltage during normal system operation.

During static IDD testing of the chip as a whole, which occurs during tester application of option release vectors, APLL bias currents are turned off under control of the ENID pin.

The canned APLL vector set, which performs tests (i)-(iv) above, toggles the ENID pin, which has no simulation model. Therefore, the customer cannot simulate these canned vectors.

Appendix A: APLL Internal Test Circuitry

Figure A.1 and Figure A.2 are more detailed versions of Figure 2 and Figure 3, respectively, showing the test control circuitry built into the APLL. Table A.1 shows how the TESTSEL and ENID signals are used to control this circuitry in order to move the APLL into each of its operating modes. The top portion of Table A.1 shows how the Motorola-internal APLL test program performs the tests described in Section 5.2.

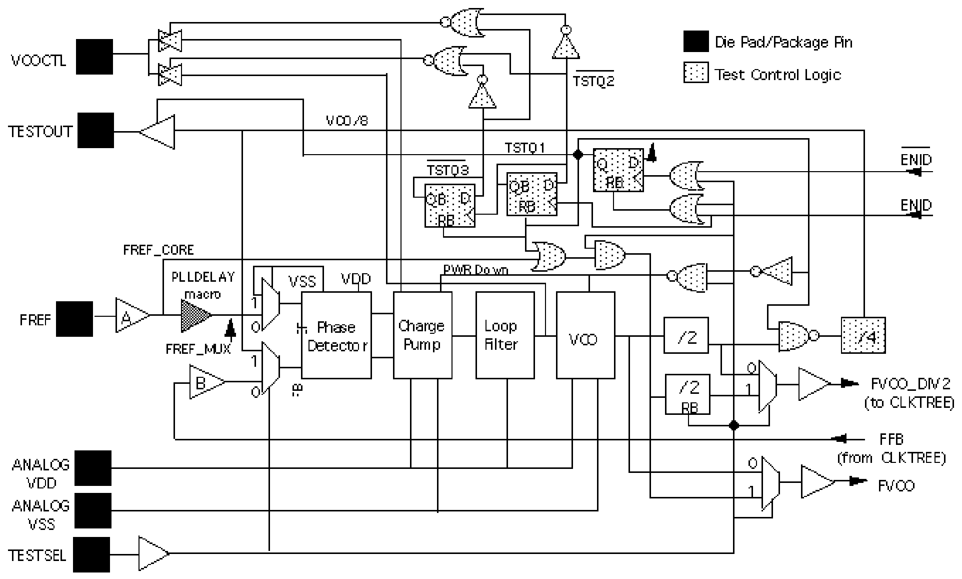


Figure A.1 Analog PLL Schematic and Test Logic (CMOS Input)

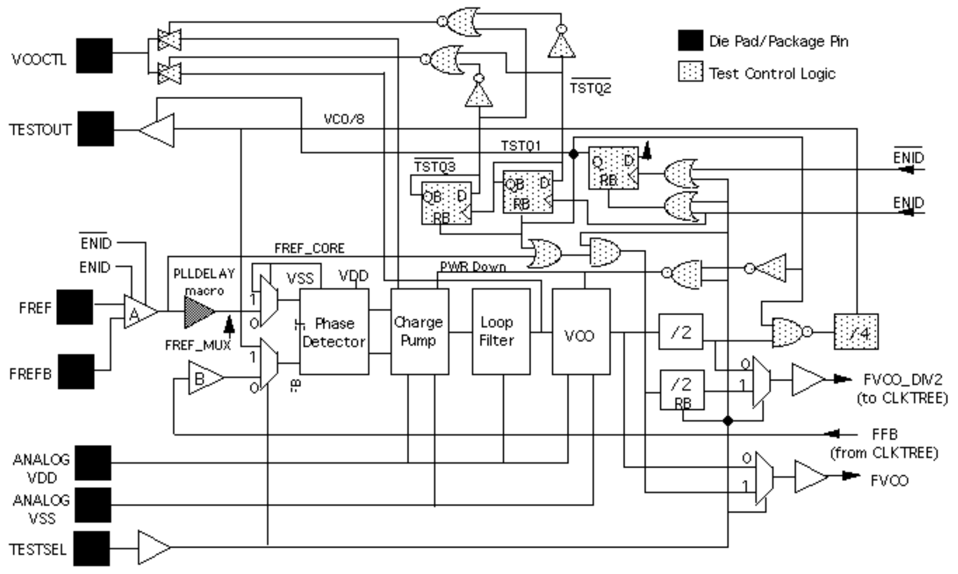


Figure A.2 Analog PLL Schematic and Test Logic (PECL Input)

Table A.1 APLL Simulation and Test Mode Sequence

Test	Inputs		APLL Internal Nodes			Simulation and Test Modes
	TESTSEL	**ENID**	**TSTQ1**	**TSTQ2**	**TSTQ3**	
APLL Test Provided at Motorola (Not User Defined)	0	0	0	0	0	**Reset test flops.**
	0	**1**	0	0	0	(Set-up Test)
	1	1	**1**	0	0	(Set-up Test)
	1	**0**	1	0	0	**Measure frequency; VCOCTL pin 3-state.**
	1	0	1	0	0	(Running Test)
	1	**1**	1	**1**	0	(Set-up Next Test)
	1	**0**	1	1	0	**Measure frequency & VCO control voltage.**
	1	0	1	1	0	(Running Test)
	1	**1**	1	**0**	**1**	(Set-up Next Test)
	1	**0**	1	0	1	**Measure frequency & charge pump current.**
	1	0	1	0	1	(Running Test)
	1	**1**	1	**1**	1	**Measure dynamic IDD of APLL.**
	1	1	1	1	1	(Running Test)
Core Test (User Defined)	**0**	**0**	**0**	**0**	**0**	**Reset test flops.**
	1	0	0	0	0	**Start functional testing of array core with APLL inactive.**
	1	0	0	0	0	(Running Test)
	1	**1**	0	0	0	**IDD vector (at Motorola only)**
	1	**0**	0	0	0	(Running Test)
	1	0	0	0	0	(Running Test)
	0	0	0	0	0	**Customer board simulation with APLL active.**

* User can only simulate states in which ENID is low/inactive.

H4CPlus, H4EPlus and M5C are trademarks of Motorola, Inc.
Verilog is a trademark of Cadence Design Systems, Inc.
Synosys is a trademark of Synopsys, Inc.

Appendix B: Transfer Functions

The APLL is a classical second order control system. Its transfer functions are:

Phase Detector Transfer Function:
$K_p = I_p/2\pi$
where I_p is the charge pump current.

Filter Transfer Function:
$K_f = R + 1/sC$
where R is the loop resistor and C is the loop capacitor.

VCO Transfer Function:
$K_o = K_v/s$
where K_v is the gain of the VCO in the linear region.

Open Loop Transfer Function:
$G(s) = K_p(K_f)(K_o)/N$
where N is the value of the divider in the feedback path.

Closed Loop Transfer Function:
$H(s) = G(s)/(1 + G(s))$
$H(s) = 2\zeta\omega_n s + \omega_n^2/(s^2 + 2\zeta\omega_n s + \omega_n^2)$
where $\omega_n = (K_v I_p/2\pi CN)^{1/2}$

Damping factor = $\zeta = RC\omega_n/2$

Typical values for the loop parameters are given in Table B.1.

Table B.1 Typical Loop Parameter Values

Parameter		3.3 V	5 V
I_p (µA)		70	100
K_o (MHz/V)		150	200
R (Ohm)	N = 1-4	2100	1400
	N = 5-16	4200	2500
C (pF)		50	50

From these typical values and the closed loop transfer function, the user can determine the characteristics of the loop and generate Bode Plots, if desired.

Appendix C: VCO Frequency vs. Voltage

Select a VCO frequency in the middle of the linear region of Figure C.1 to optimize damping.

Example: Using a 5V APLL, if the desired clock tree frequency (CLK) is 50 MHz (see Figure 10) and FREF is 25 MHz, select "/M" = 2 and "/L" = 2. This gives CLK = 50 MHz, VCO_DIV2 = 100 MHz and VCO = 200 MHz which is in linear region of Figure C.1.

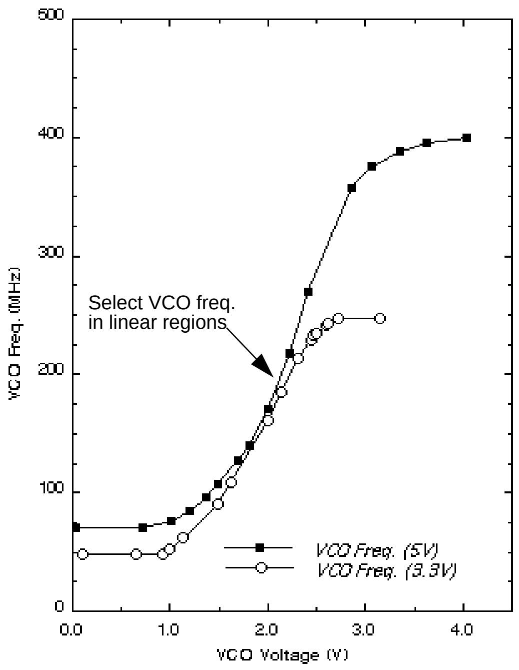


Figure C.1 VCO Frequency vs. Voltage

Table C.1 Minimum Operating Frequency

Divide Factor, N	3.3 V			5 V		
	Filter Resister, R	Damp-ing, ζ	F_{pdmin} (MHz)	Filter Resister, R	Damp-ing, ζ	F_{pdmin} (MHz)
1	2100	0.76	66	1400	0.70	84
2	2100	0.54	33	1400	0.49	42
3	2100	0.44	22	1400	0.40	28
4	2100	0.38	17	1400	0.35	21
5	4200	0.68	26	2800	0.63	34
6	4200	0.62	22	2800	0.57	28
7	4200	0.58	19	2800	0.53	24
8	4200	0.54	17	2800	0.49	21
9	4200	0.51	15	2800	0.47	19
10	4200	0.48	13	2800	0.44	17
11	4200	0.46	12	2800	0.42	15
12	4200	0.44	11	2800	0.40	14
13	4200	0.42	10	2800	0.39	13
14	4200	0.41	9	2800	0.37	12
15	4200	0.39	9	2800	0.36	11
16	4200	0.38	8	2800	0.35	11
Notes:	Ip=0.07mA, C=50 pF, VCO gain =1.5x10^8 MHz/V			Ip=0.1mA, C=50 pF, VCO gain =2x10^8 MHz/V		

Appendix D: PLL Basics

(From application note "ASIC Distribution Using a Phase Locked Loop (PLL)", AN1509)

D.1 INTRODUCTION

Transferring data between ASIC chips at frequencies above 40 MHz requires special on-chip circuitry in current sub-micron technologies. Phase locked loops can provide skew management in ASIC devices to help compensate for clock-tree insertion delays and process, temperature and voltage variations allowing maximum multi-chip system performance.

This application note is written to help designers of multi-chip ASIC systems maximize system performance by managing clock distribution and optimizing clock skew and data path relationships. It contains equations relating measurable timing and skew parameters to maximum frequencies of operation. It explains techniques available to minimize critical parameters which contribute to clock skew.

D.2 BACKGROUND

D.2.1 REGISTER-TO-REGISTER DATA TRANSFER BETWEEN ASIC CHIPS

When determining the maximum frequency at which data can be transferred from one ASIC device to another, a designer must carefully consider both the delay of the data path and the skew of the clock. The data path is the delay from a register in the sending ASIC (including clock to Q) to the D input of a register in the receiving ASIC (including the setup and hold times), see Figure D.1. The clock skew or Tskew is the difference between a rising edge on ClkA in ASIC1 and ClkB in ASIC2.

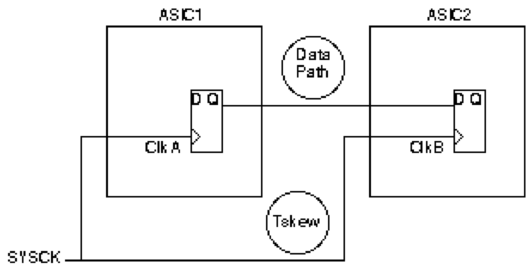


Figure D.1 Chip-to-Chip Timing Parameters

Tskew in this document refers to clock skew in both the positive and negative directions. Positive skew is when the rising edge of ClkB occurs later than a rising edge of ClkA. Positive skew affects data transfer from a hold time standpoint. Negative skew is when the rising edge of ClkB occurs earlier than a rising edge on ClkA. Negative skew affects data transfer from a setup time standpoint. A complete analysis of clock skew is performed in Appendix E.

D.2.2 SETUP AND HOLD TIME CONSIDERATIONS

To insure error-free data transitions between ASIC1 and ASIC2, the data path from the sending flip- flop in ASIC1 to the receiving flip flop in ASIC2 must not be so long that a setup time violation is realized on the receiving flop- flop. The same data path must also be long enough to avoid a hold-time violation on the receiving flip-flop. This setup and hold time relationship must take into consideration clock skew between the rising edge of ClkA, which initiates the data transfer and the rising edge of ClkB which clocks in the transferred data.

D.2.3 INSERTION DELAY AND THE EFFECT OF THE CLOCK-TREE

Insertion delay is defined as the delay from the rising edge of the external system clock to the rising edge of the clock on any given flip-flop on the ASIC. In Figure D.2, it's the delay from SYSCK to ClkA or ClkB. Insertion delay is made up of the clock input buffer and clock-tree delays. The insertion delay in one ASIC can be very different from the insertion delay in another ASIC, depending on the size of the ASIC and the number of elements that must be clocked by the cloc- tree. Differences in insertion delays between ASIC devices directly contribute to clock skew (Tskew). In the example circuit (Figure D.2): if ASIC1 has an insertion delay of 5 ns and ASIC2 an insertion delay of 10 ns, then a rising edge in ASIC 1 will be skewed by at least 5 ns from a rising edge in ASIC 2.

D.2.4 PTV VARIATIONS

Process, Temperature and Voltage (PTV) variations can increase the difference in insertion delays. Most ASIC technologies use a multiplier to adjust delays due to PTV. In the H4C technology, a worst-case multiplier (WCM) and a best-case multiplier (BCM) are used. The WCM modifies a typical delay to represent worst-case conditions. The WCM is greater than one. The BCM is less than one and modifies a typical delay to represent a best-case condition. The "process spread" is the difference between a best-case delay and a worst-case delay for a given data path. The process spread can be found by dividing the WCM by the BCM (WCM/BCM). Choosing a technology with a minimum process spread will allow higher overall performance.

D.2.5 MAXIMUM FREQUENCY OF OPERATION

An equation can be derived that relates setup and hold times, insertion delay and process spread to determine the maximum frequency at which data can be safely transferred from chip-to-chip. A full derivation of this equation is provided in Appendix E. The equation in terms of the minimum period is,

$$MinPer=Tskew(WCM/BCM+1)+WCM(Tsu+Th+TDm) \quad (D.1)$$

where,

MinPer	Minimum clock period in ns (1/max frequency of operation).

Tskew	Total skew (positive and/or negative) between rising edges of ClkA and ClkB (see Figure D.2).
WCM	Worst Case Multiplier.
BCM	Best Case Multiplier.
Tsu	Setup delay of flip flop in receiving ASIC (ASIC2 in Figure D.2).
Th	Hold delay of flip flop in receiving ASIC (ASIC2 in Figure D.2).
TDm	Data path delay margin

Two things become apparent in looking at Equation (D.1). First, Tskew is the dominant parameter affecting the maximum frequency at which data can be transferred between ASIC devices. Secondly, the process spread for the chosen technology is also very important. Clearly, Tskew and the process spread must be minimized to allow maximum performance.

To address the problem of clock skew, a Phase Locked Loop (PLL) can be added to each ASIC device to reduce the effects of insertion delay differences and help manage the skew from chip-to-chip. The PLL will synchronize the rising edge on SYSCK such that it will be simultaneous with a rising edge on flop ck, see Figure D.3. If the PLL is used on each ASIC device, all flop ck signals on every ASIC will be simultaneous within the error of the PLL. The PLL will compensate for differences in insertion delays from ASIC-to-ASIC as well as PTV variations.

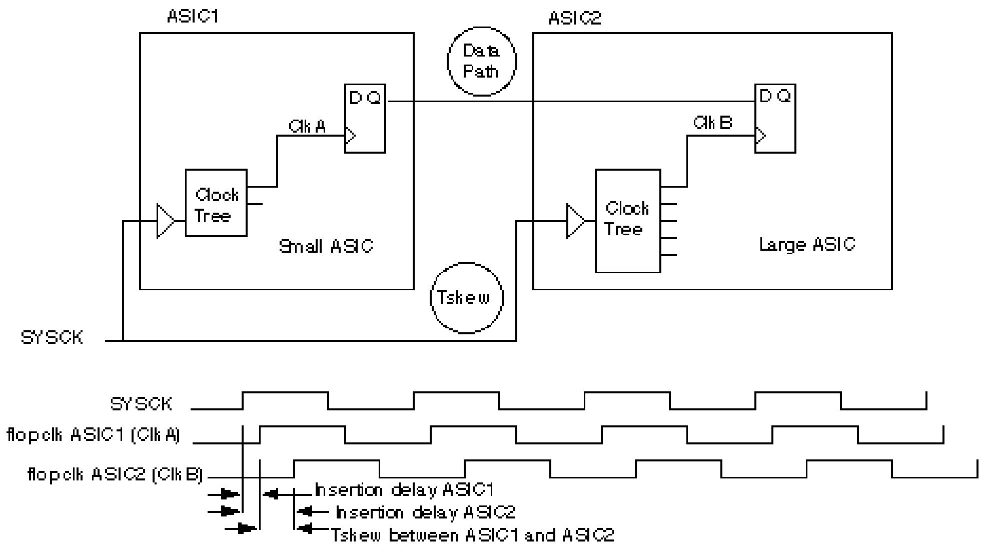


Figure D.2 Effect of Clock Tree on T_{skew}

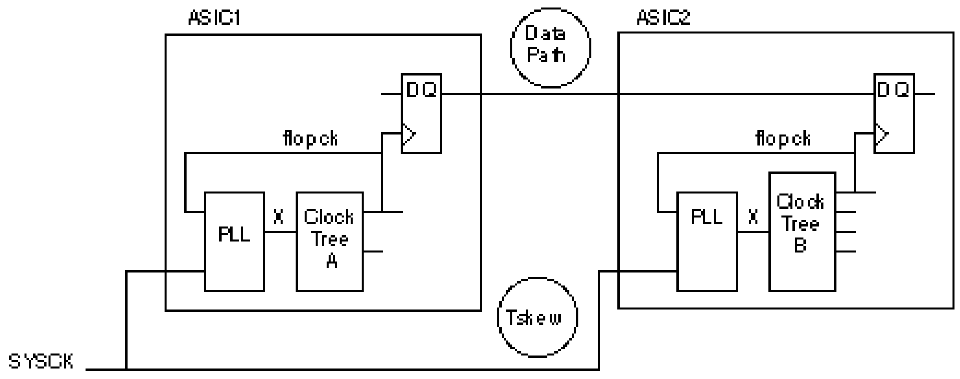


Figure D.3 PLL Solution

Appendix E: Derivation of Minimum Period Equation

(From application note “ASIC Distribution Using a Phase Locked Loop (PLL)”, AN1509)

This section contains a derivation of the equation that relates clock skew, process spread, and flip-flop specifications to determine the minimum period or maximum frequency at which data can be transferred between ASIC devices. Figure E.1 illustrates the data and clock paths between two ASICs. If data is to be transferred reliably from ASIC1 to ASIC2, the set up and hold time requirements of the receiving flip flop in ASIC2 must be satisfied in the presence of clock skew and process spread. First, we will analyze the setup and hold time requirements of the receiving flip flop. This is similar to the classic shift register problem where clock skew can cause setup or hold problems on the receiving flip flop.

The data delay path from ASIC1 to ASIC2 includes 1) the delay from a rising edge of ClkA to the output of ASIC1 - TDout, 2) the delay of the PC board trace - TDbrd and 3) the delay of the input path of ASIC2 - TDin. The setup and hold time parameters of the receiving flip flop Tsu and Th must also be considered. The total data path delay is,

$$TD = TDout + TDbrd + TDin \qquad (E.1)$$

When considering the setup time requirements of ASIC2, the worst case path from ASIC1 to ASIC2 must be considered. The minimum period at which data can be safely transferred in the presence of clock skew without violating the setup requirements of the receiving flip flop is,

$$MinPer = WCM(TD + Tsu) + Tskew \qquad (E.2)$$

Note that typical delay values are used in these equations. These values are modified for best case and worst case by the multipliers BCM and WCM respectively. Additionally, the worst- case path assumes the edge direction, rising or falling that results in the longest delay.

Figure E.2 illustrates the setup time requirement. The dashed lines on ClkB represent clock skew.

When considering the hold- time requirements of ASIC2, the best-case path must be considered. The best-case path assumes the edge direction, rising or falling that results in the shortest delay. The equation relating the data path, hold time and Tskew is,

$$BCM(TD) \geq Tskew + BCM(Th) \qquad (E.3)$$

We now have two equations relating the data path. To find the minimum period, first consider the ideal case, then generalize it. Ideally, assume the data path delay TD is just long enough to prevent a hold-time violation, or the best-case datadelay is equal to the hold time plus the clock skew,

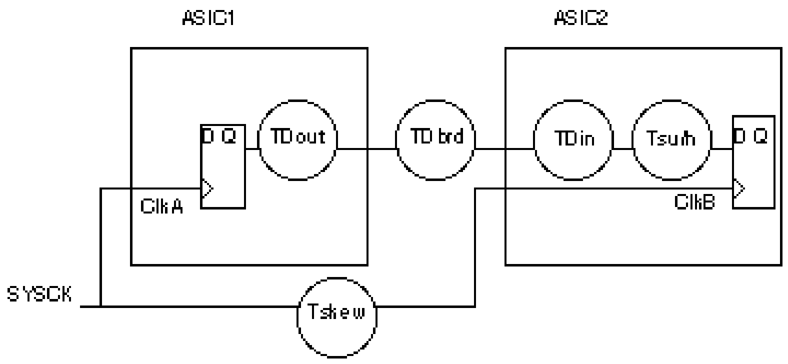


Figure E.1 Chip-to-Chip Data Transfers

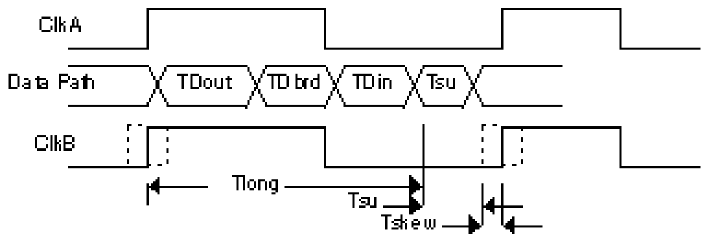


Figure E.2 Clock Skew and Setup Time Requirements

BCM(TD) = Tskew + BCM(Th) (E.4)

If true, we could solve this equation for TD and put that value into the setup time equation,

TD = (Tskew+BCM(Th))/BCM (E.5)

MinPer = WCM(((Tskew+BCM(Th))/BCM) + Tsu) + Tskew (E.6)

Notice that Tskew appears twice in this equation. Skew in the positive direction affects hold time and skew in the negative direction affects the setup time. Generally, we don't know if the clock skew is in the positive or negative direction so we consider it twice.

Figure E.4 illustrates this equation. The minimum period is found by taking the best-case data path delay that is just long enough to prevent a hold-time violation in the presence of clock skew, BCM(TD), multiply that delay by the worst-case multiplier WCM(TD), and add to that the worst-case setup time and the clock skew.

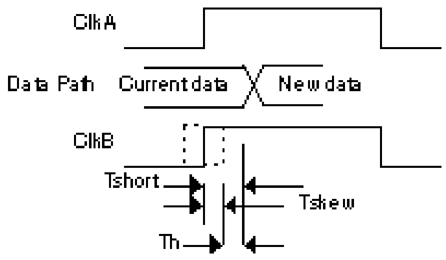


Figure E.3 Clock Skew and Hold Time Requirements

Equation (E.5) can be reduced to become very close to our final equation,

MinPer = WCM (Tskew/BCM + Th + Tsu) + Tskew

MinPer = WCM/BCM (Tskew) +WCM(Th +Tsu) + Tskew combining the Tskew terms,

MinPer = Tskew (WCM/BCM+1) + WCM (Tsu+Th) (E.7)

It is unrealistic to assume all data paths can be tuned to be just long enough to prevent a hold-time violation. We should therefore introduce some margin in the data path. Generally, this margin would be defined by the shortest chip-to-chip data path delay on one end of the spectrum and the longest chip-to-chip data path delay on the other end. This assumes, of course, that these paths are long enough or short enough to prevent hold time or setup time violations respectively. When designing shift registers from discrete components, it is common to add delay to the data path to insure there is not a hold time violation in the presence of clock skew. If the maximum frequency of a system is limited by the data path delay from chip-to-chip (see Equation (E.2)) it may be necessary to add delay to shorter data paths to prevent hold times. There should always be a 1-2 ns margin (typical delays) between the shortest data path and the longest data path to allow room for variation in delay as the paths are tuned to prevent violations. If this margin TDm is added, a new equation and timing diagram result. From Equation (E.5) we add the margin TDm to the typical data delay TD,

TD = ((Tskew+BCM(Th))/BCM) + TDm (E.8)

The minimum period is,

MinPer=WCM((((Tskew+BCM(Th))/BCM)+TDm)+Tsu) +Tskew

MinPer = WCM(Tskew/BCM + Th +TDm + Tsu) + Tskew

MinPer=WCM/BCM(Tskew)+WCM(Th+TDm)+Tsu) +Tskew

MinPer = Tskew (WCM/BCM + 1) + WCM (Tsu + Th + TDm) (E.9)

Note that if the period of operation is larger than the minimum defined above, the data path margin TDm will be larger and there will be more room for data path tuning.

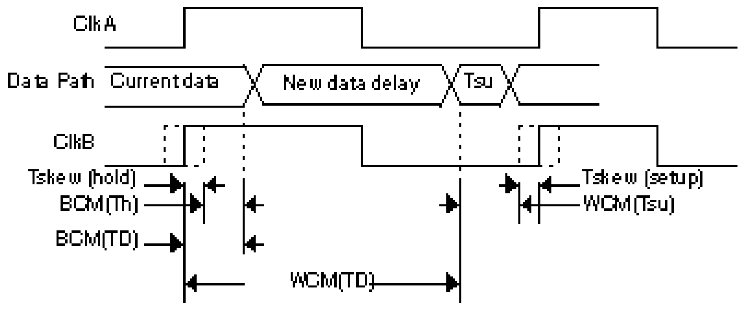


Figure E.4 Ideal Minimum Period Considering Setup and Hold Time

Notes:

ASIC REGIONAL DESIGN CENTERS - U.S.A.

California, San Jose (408) 749-0510	Illinois, Chicago (708) 490-9500	Massachusetts, Marlborough (508) 481-8100

ASIC REGIONAL DESIGN CENTERS - International

European Headquarters, Germany, Munich (089) 92103-0	England, Aylesbury, Bucks (0296) 395252	France, Vanves (01) 40355877
Holland, Eindhoven (04998) 61211	Hong Kong, Kwai Chung 480 8333	Italy, Milan (02) 82201
Japan, Tokyo (03) 440-3311	Sweden, Stockholm (08) 734-8800	

Literature Distribution Centers:

USA:	Motorola Literature Distribution; P.O. Box 20912, Phoenix, Arizona 85036
EUROPE:	Motorola Ltd.; European Literature Center; 88 Tanners Drive, Blakelands, Miltion Keynes MK14 5BP, England
JAPAN:	Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan
ASIA-PACIFIC:	Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong

MOTOROLA