By Anis Jarrar, Kelvin McCollough, Jim Caserta, Motorola

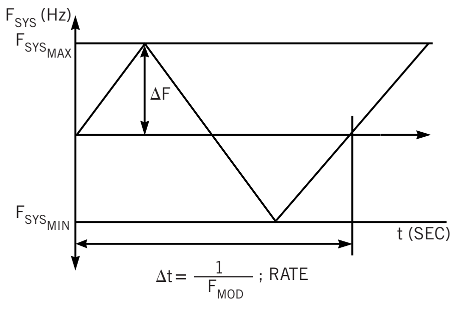
PLL CIRCUITS OFFER DEVELOPERS OF MIXED-SIGNAL DESIGNS A UNIQUE CHALLENGE. COMMONLY USED AS FREQUENCY SYNTHESIZERS, CLOCK MULTIPLIERS, OR CLOCK-RECOVERY DEVICES, PLLS ARE CRUCIAL TO MANY MICROCONTROLLERS AND MICROPROCESSORS. AS MORE FUNCTIONS ENTER THE DIGITAL DOMAIN, VERIFICATION BECOMES MORE DIFFICULT AND REQUIRES MORE SOPHISTICATED MODELS. A VERILOG-ONLY MODEL CAN SOLVE MOST OF THESE PROBLEMS.

Simulating mixed-mode designs with Verilog-only models

As the capacities of today's chips grow and their geometries shrink, specialized clock-generation circuits are necessary to reduce electrical-magnetic emissions. One scheme that is gaining industry acceptance is frequency modulation of the system clock. Validating such systems has been notoriously difficult in the past, primarily because of the extended simulation times that validating required. As a result, high-level models became a necessity for design validation. Engineers usually developed such models in C; Fortran; or a commercial mathematical tool, such as Matlab. Yet, modern simulators and computing resources have become advanced enough to allow transistor-level verification of PLLs. Recently, designers have used tools, such as Synopsys' PowerMill ACE (Analog Circuit Engine), to verify the PLLs at the transistor level. PowerMill ACE was used to verify the first generation of FM-PLLs on earlier microcontroller chip designs.

Validating the second generation of FM-PLLs requires a different strategy, because most of the new development is in the digital control logic. Designers developed and implemented a calibration algorithm to trim the magnitude of the change in frequency superimposed on the system clock. The calibration is necessary to overcome the limitations that the baseline process and the large variation in the analog building blocks impose. Designers can resolve the final change in frequency to within 2% of the system frequency with a maximum 2% error. The calibration sequence requires approximately 1 msec in real time. Prior to adding the calibration, the PLL achieved frequency lock in approximately 200 msec in real time. Therefore, the total required real-time simulation is approximately 1.2 msec, and consequently, the verification emphasis shifts from analog to digital.

Figure 1



NOTES:

$F_{SYS_{MAX}} = F_{SYS} + \{2\%, 4\%, 6\%\}$.

$F_{SYS_{MIN}} = F_{SYS} - \{2\%, 4\%, 6\%\}$.

$F_{MOD} = \frac{F_{REF}}{Q}$, WHERE Q = {40,80}.

A modulation waveform illustrates frequency of the clock generated by FM-PLL over time.

In response to this strategy, designers developed a Verilog-only model, allowing the rapid and extensive verification of the new calibration algorithm. Although the ACE option is still valid, its simulation time is prohibitive. Using a Verilog/VerilogA model is another option, but models are not fully developed. Moreover, fully developed VerilogA models could increase the simulation times to the point at which using the ACE becomes a better option. This situation is especially true when you consider that you need not develop special models to support ACE. VerilogA is a good tool for architecture trade-off analysis and for conducting what-if scenarios at the beginning phases of the design process.

THE PLL DESIGN

FM-PLLs use a reference of 2 to 20 MHz to produce an output range of 80 to 160 MHz. The system that incorporates this PLL operates at one-quarter of the FM-PLL's output-clock frequency. The frequency modulation targets a 100- or 200-kHz rate and a depth of 2, 4, or 6% of the system frequency (**Figure 1**).

The digital portion of the PLL is synthesized. Its functions include a feedback divider, lock detection, a center-frequency search algorithm, and a frequency-modulation algorithm. The digital block uses about 2165 cells. The total RTL (register-transfer-level) count reaches 1669 lines. The nominal power-supply voltage is 3.3V, and the targeted power consumption is less than 2 mA for the total analog circuitry. The designers used a CDR1 (Communications Design Rules) baseline process of 0.35-mm Leff, 90A oxide threshold voltages targeting a 600-mV, single-polysilicon, triple-metal CMOS process. The PLL comprises a phase-frequency detector, charge pumps, two RC loop filters, main and reference ICOs (current-controlled oscillators), a current-reference generator, a voltage reference, a frequency-to-current converter, current D/A converters, divider blocks Q and P, center current SAR (successive approximation register), calibration control, a calibration shift register, and loss-of-lock and loss-of-clock detection (**Figure 2**).

THE FM-PLL VERILOG MODEL

The BLM (behavioral-level model) of an FM-PLL aims to provide an accurate model of all the analog blocks within the module, enabling you to verify the increasingly complex digital section using a standard logic simulator, such as Verilog-XL. The BLM maintains the existing design hierarchy, reuses all of the RTL code for the digital blocks without modification, and keeps the I/O list for each module the same as the real

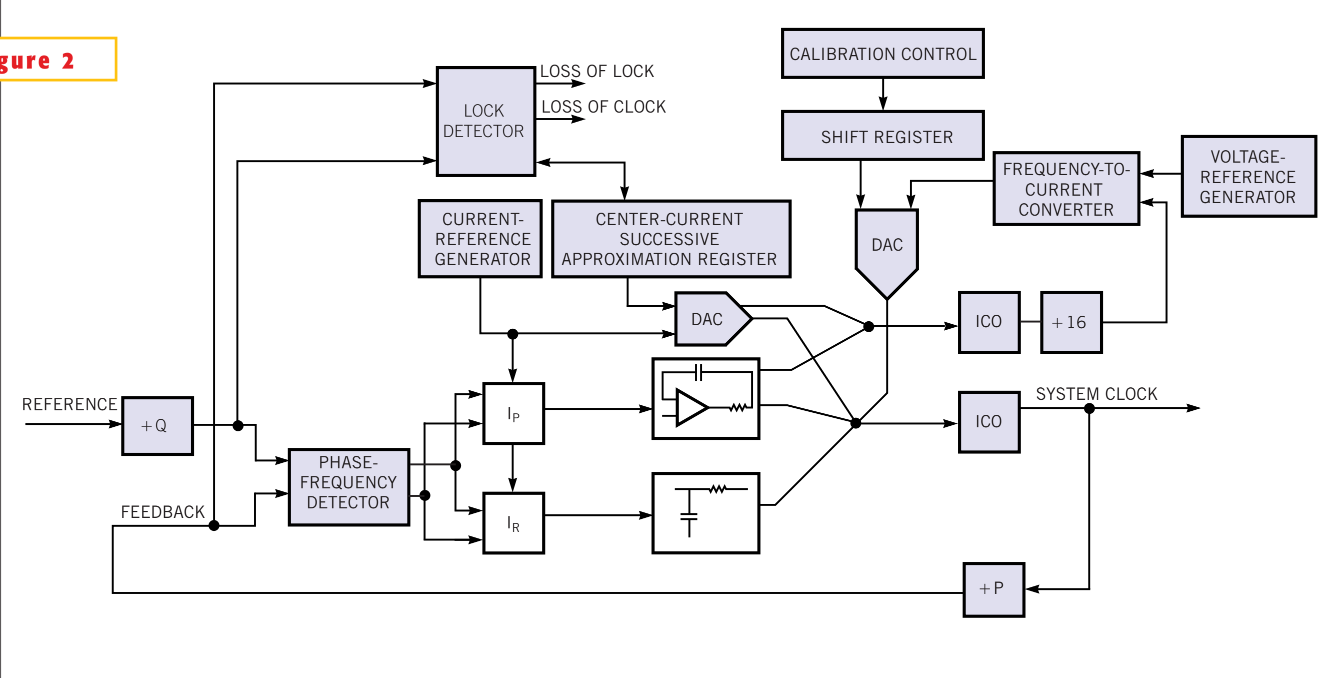
LISTING 1—PARAMETRIC DEFINES FILE

```
'define SAMPLE_FREQ 2.5e9 // Sample Clk Freq. for discrete sim.
'define VDD_SYN_VAL 3.300 // Synthesizer Supply Voltage
'define ICO_CK 454.5e-15 // Capacitor in ICO gain circuit
'define FTOI_C 228e-15 // Capacitor in FTOI circuit
'define CHGPMP_IR 8.0e-6 // Bias Current in IR Charge Pump
'define CHGPMP_IP 1.6e-6 // Bias Current in IP Charge Pump
'define FILTER_C3 10.0e-12 // Capacitor Value in IR Filter circuit
'define FILTER_R3_FM 132.0e3 // Resistor Value in IR Filter
'define FILTER_CINT 80.0e-12 // Capacitor Base Value in IP Filter
'define FILTER_RT 12.0e3 // Resistor Value in IP Filter circuit
```

LISTING 2—I_P VERILOG MODEL

```
initial
// Voltage va at node should be at ICO_VREF at time 0
va = 'ICO_VREF;
va_prev = va;
filter_ip_out = 0.0;
filter_cint_value = 'FILTER_CINT;
end
always @(posedge 'SAMPLE_CLK)
begin
if(pll_pintenable)
begin
va_prev = va; // assign current value to previous value
va = va_prev + ('PLL_SCOPE.IR.chgpmp_ip_out /
(filter_cint_value * 'SAMPLE_FREQ));
filter_ip_out = (va - 'ICO_VREF)/'FILTER_RT;
end
end
```

Figure 2



An FM-PLL block diagram details the various analog and digital blocks in the design.

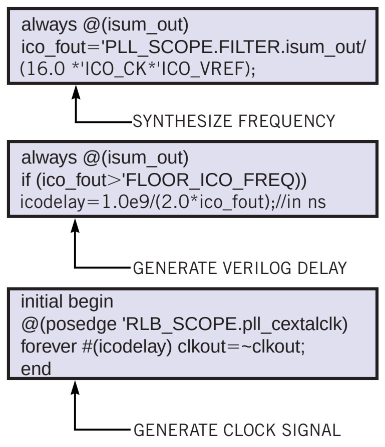
model. Several challenges arise in creating an accurate BLM; most are due to Verilog's inherent inability to model and simulate currents and voltages.

To overcome these obstacles, you should create a virtual high-frequency clock to perform discrete time simulation. The simulator treats all voltage and current values as real Verilog variables and samples and updates their values on the positive edge of the virtual clock. The clock provides the necessary time increment (Δt) to update the critical currents and voltages within the analog circuits that allow the BLM to accurately model the real circuit. The virtual clock for this simulation has a frequency of 2.5 GHz and provides the simulator a time increment of 400 psec. This value depends on the bandwidth of the analog components you are modeling and the resolution that the simulation requires. Take care in selecting this frequency; arbitrarily increasing it will negatively impact the overall simulation time.

A method for passing current and voltage values between the blocks is necessary to complete the BLM. One possible method is to convert these values into vectored signals between the various modules. However, because the I/O list for each block must remain the same to eliminate the need for a separate simulation database, this method is inadvisable. Instead, you should use hierarchical variable references and global defines to achieve the required block connectivity.

This model generates a frequency-modulated clock that can interact with and that you can simulate with the rest of the digital logic in the module without a computation-intensive mixed-mode simulator. To ease the reuse of this module, designers created a Verilog-environment file that defines all critical circuit parameters (some of which you obtain through traditional Spice simulations). This file is included with the rest of the Verilog design files, and you can change it at the beginning of every simulation to explore design trade-offs. **Listing 1** contains an excerpt from this file.

Figure 3



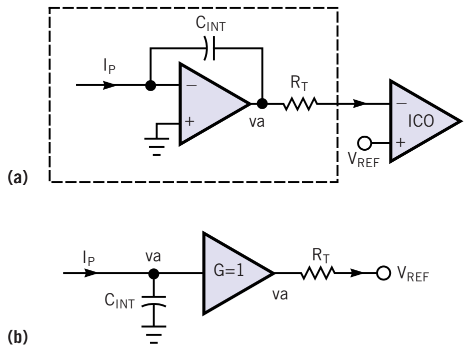
```
always @(isum_out)
ico_fout='PLL_SCOPE.FILTER.isum_out/
(16.0 *'ICO_CK*'ICO_VREF);
```

```
always @(isum_out)
if (ico_fout>'FLOOR_ICO_FREQ))
icodelay=1.0e9/(2.0*ico_fout);//in ns
```

```
initial begin
@(posedge 'RLB_SCOPE.pll_cextalclk)
forever #(icodelay) clkout=~clkout;
end
```

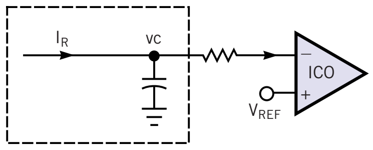
Verilog code describes how the system clock period is generated.

Figure 4



Original (a) and simplified (b) I_P loop-filter equivalency models show how you can simplify the model to facilitate modeling within Verilog.

Figure 5



You can model an I_R loop filter as a simple RC circuit.

BLOCKS MODELED IN VERILOG

The relationship between the current and the frequency in the main oscillator is linear. The product of the ICO gain and the current that is supplied to the oscillator determine the frequency. This assumption is a simplified approximation of the ICO operation, because frequency shifts in this model occur instantly in response to input current changes. Once you know the ICO frequency, you can synthesize the system clock by defining the clock high and low times. **Figure 3** illustrates how you can perform this calculation in Verilog.

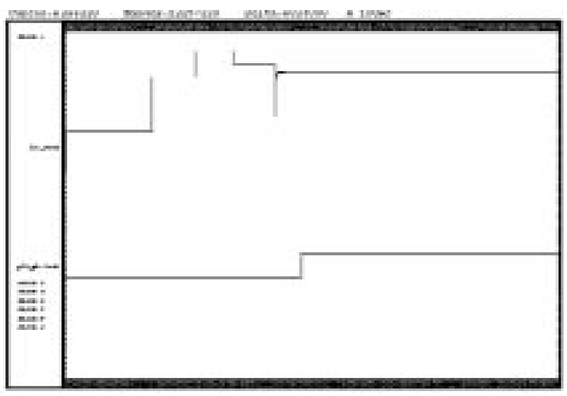
The I_P charge-pump loop filter is modeled as an integration capacitor (C_{INT}) connected to a unity gain buffer whose output connects to a filter resistor (**Figure 4**). The current entering the ICO from the filter is modeled in Verilog (**Listing 2**). The I_R charge-pump loop filter is modeled similarly to the I_P loop filter as a simple RC circuit (**Figure 5**). The current entering the ICO from the filter is modeled in Verilog as shown in **Listing 3.**

The PFD (phase-frequency detector) is a custom digital circuit that generates the up-and-down control signals to the charge pumps. These signals control the polarity of the charge pumps' output currents. **Listing 4** shows both the PFD and charge-pump models.

The frequency-to-current circuit generates the necessary current to perform the frequency dithering of the main ICO output clock. It uses a 2-bit signal to control the modulation depth. The output current is proportional to the input frequency and reference voltage. An 8-bit DAC converter supplies this reference voltage. The Verilog code in **Listing 5** implements the function described above.

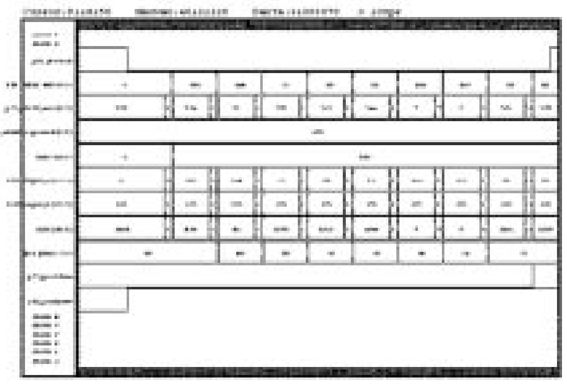
The simulation testbench integrates

Figure 6



Behavioral-level model current-controlled-oscillator frequency shows the PLL generating the expected frequency during and after calibration.

Figure 7



Register values in the calibration block change during the calibration sequence to control frequency modulation.

the analog BLM and the digital control blocks into the microcontroller and simulates them in the full-chip Verilog environment. Verification engineers write nine patterns to test the FM-PLL digital block. Engineers also run 30 legacy patterns to verify basic PLL functions. The patterns verify the digital-control RTL and the gate netlist. Initially, verification engineers created vcd (value-change-dump) files so that designers could examine the simulation output to verify proper BLM operation and basic calibration functions. All patterns generate log files detailing the pass/fail status of each operation. A typical pattern contains the following steps:

1. Enter PLL Mode; wait for PLL to lock. Using the vcd file, you can observe the frequency of the output clock and compare it with the expected behavior. **Figure 6** shows the ICO clock frequency and lock status bit.

2. Check lock-bit status. The lock bit is set only when Ffdbk=Fref, thus you can measure the frequency and time at which the lock bit is set. You determine the frequency at which the PLL locks using the vcd file and verify the lock-bit status with the pattern.

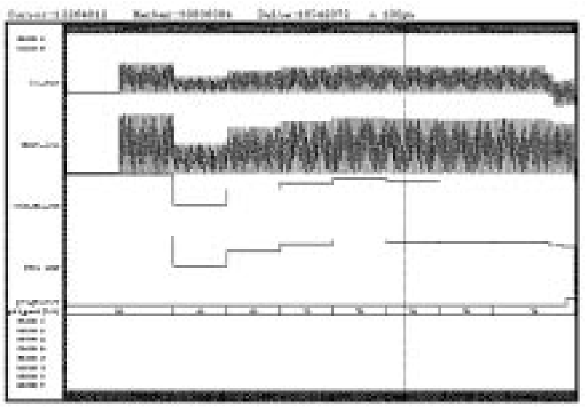
3. Program modulation by writing to specific memory locations. This action initiates the calibration sequence.

4. Observe calibration routine. The ICO clock in the BLM is modulated to provide the stimulus for the digital control block. You can observe intermediate digital values by the pattern for each iteration of the calibration routine, including the measured frequency count, the frequency offset count, the difference between actual offset and target offset, the calibration DAC code, and a calibration-status bit. The analog effects you observe in the vcd file are the change in ICO frequency, the frequency-to-current output current, the modulation DAC output current, and the calibration DAC output voltage (**figures** 7 and **8**).

5. Wait for calibration routine to end. The pattern checks the calibration completion flag.

6. Wait for PLL to relock. You observe the frequency shift in the

Figure 8



Analog outputs change during the calibration sequence in response to control signals from the calibration block.

LISTING 3—I_R VERILOG MODEL

```
initial
begin
vc = 0.0;
vc_prev = 0.0;
filter_ir_out = 0.0;
end
always @(posedge 'SAMPLE_CLK)
begin
if(pll_pintenable) // if filter is enabled
begin
vc_prev = vc;
filter_ir_out = ('PLL_SCOPE.IR.chgpmp_ir_out +
('FILTER_C3 * 'SAMPLE_FREQ * (vc_prev - 'ICO_VREF)))
/ (1.0 + 'FILTER_C3 * 'SAMPLE_FREQ * filter_r3_value);
vc = vc_prev +
(('PLL_SCOPE.IR.chgpmp_ir_out - filter_ir_out) /
('FILTER_C3 * 'SAMPLE_FREQ));
end
end
```

vcd file as the loop closes again. You measure the change in frequency from the vcd file after the calibration routine finishes. The pattern reads the final calibration settings.

Verification patterns are developed for many purposes; general patterns observe operation for different register settings (multiplication factor, or divP; modulation depth; or modulation rate). You can model a process shift by modifying the environment file, thus introducing errors into the BLM. You can also model voltage or temperature shifts in a similar fashion to produce errors in the model. Running a general pattern allows you to observe the calibration routine and see how it trims out the error by adjusting analog values in the BLM and thus produces the desired change in frequency. Aligned and unaligned patterns verify proper operation of the control block's arithmetic logic unit. Calibration-self-check patterns verify the proper operation of the calibration-status bit. The patterns check that the bit is set while calibration is enabled, unless the pattern induces a failure. When a failure is induced or calibration is disabled, you should clear the calibration-status bit. Overflowing the counters by generating a clock frequency greater than the PLL specification is one way to induce failure. Make control-register changes at different stages of the algorithm to observe proper state-machine operation.

LISTING 4—PHASE-FREQUENCY DETECTOR AND CHARGE-PUMP MODELS

```
always @(posedge refclk or posedge reset_updown)
begin
if(reset_updown)
up_w = 0;
else
up_w = 1;
end
always @(posedge fdbkclk or posedge reset_updown)
begin
if (reset_updown)
down_w = 0;
else
down_w = 1;
end
always @(up_w or down_w)
begin
if (up_w & down_w)
begin
#2 reset_updown = 1;
#1 reset_updown = 0;
end
// The real values of the Ir/Ip currents going to the ICO are
// update on the rising edge of the sample clock
always @(posedge ‘SAMPLE_CLK)
begin
chgpmp_ir_out = ‘CHGPMP_IR * (up_val - down_val);
chgpmp_ip_out = ‘CHGPMP_IP * (up_val - down_val);
end
```

LISTING 5—FREQUENCY-TO-CURRENT CONVERTER MODEL

```
// ccontrolmsb signals are mask plug options in the F/I circuit.
always @(‘PLL_SCOPE.MAINICO.ico_fout or ‘FTOI_VREF
or moddepth or en)
begin
if (en)
ftoi_iout = (0.25 + (0.25 * ccontrolmsb[2]) +
(0.25 * ccontrolmsb[1]) + (0.25 * ccontrolmsb[0])) *
(1.0/4.0)*‘FTOI_C*‘FTOI_VREF*
(‘PLL_SCOPE.MAINICO.ftoi_ico_fout/16.0) *
((moddepth[0] * 1.0) + (moddepth[1] * 2.0));
else
ftoi_iout = 0.0;
end
```

A lock, calibration, and relock sequence runs in 30 minutes. For RTL debugging, this length of time is a limiting factor. For debugging the gate-level netlist and timing, you should perform synthesis and place and route after all RTL code or constraint changes. This process takes approximately four hours. Designers completed the RTL and timing analysis of the FM-PLL design in six weeks, allowing timely delivery. By comparison, a concatenation of one lock, calibration, and re-lock sequence running in ACE takes five days to complete. Thus, the design team was able to achieve a 240X reduction in simulation time. The designers perform all simulations on Ultra30 Sun workstations (300 MHz with 1024 Mbytes of RAM). The number of RTL code iterations needed for the digital-control-logic debugging would be impossible if the designers used ACE as the verification tool. Using a compiled Verilog simulator instead of an interpretive simulator (Verilog-XL) reduces the time by a factor of five, bringing simulation time to less than 5 minutes.

The BLM highlights inadequacies that occur when you use unit delays for gate-level simulation. In the BLM, the ICO generates a frequency of 60 MHz, which feeds a post divider. This divider, which is characterized in silicon, can handle signals greater than 100 MHz; however, the gate model is stuck at 0 for any frequency greater than 60 MHz.

Using a Verilog simulation environment dramatically increases the number of people who can verify the module. The model allows insight into analog functioning using a pure digital simulator. Test engineers port these PLL verification patterns to the Teradyne J971 tester, and manufacturing uses them in production. Testing the calibration routine in production increases the level of testing of the analog circuits.

BLMs allow the FM-PLLs to improve their accuracy by adding digital functions. Processing, voltage, and temperature variations limit the accuracy of the analog circuitry. BLMs allow rapid and extensive test coverage and let the digital-verification team assist in verifying mixed-signal modules. You can measure the benefits of BLMs and verification in your ability to meet mask-shop deadlines and achieve first-pass silicon functioning within specification.□

AUTHORS' BIOGRAPHIES

Anis Jarrar is a principal engineer and a senior digital designer at the Motorola Microcontroller Division in Austin, TX. He holds both a BS and an MS in electrical engineering from the Georgia Institute of Technology (Atlanta).

Kelvin McCollough is a member of the technical staff and a senior analog designer at Motorola's Advanced Vehicle Systems Division in Austin, TX. He is a graduate of the University of Illinois (Urbana-Champaign). He also holds an MS in electrical engineering from the University of Missouri (Columbia).

Jim Caserta is an analog designer at the Motorola Microcontroller Division in Austin, TX. He is a graduate of the University of Florida (Gainesville), where he is currently pursuing his PhD in electrical engineering.