A Video Driver System Designed Using a Top-Down, Constraint-Driven Methodology

Iasson Vassiliou, Henry Chang†, Alper Demir, Edoardo Charbon†, Paolo Miliozzi, Alberto Sangiovanni-Vincentelli
University of California, Berkeley CA, USA. †Cadence Design Systems, San Jose CA, USA.

Abstract

To accelerate the design cycle for analog and mixed-signal systems, we have proposed a top-down, constraint-driven design methodology. The key idea of the proposed methodology is hierarchically propagating constraints from performance specifications to layout. Consequently, it is essential to provide the necessary tools and techniques enabling the efficient constraint propagation. To illustrate the applicability of the proposed methodology to the design of larger systems, we present in this paper the complete design flow for a video driver system. Critical advantages of the methodology illustrated with this design example include avoiding costly low level re-designs and getting working silicon parts from the first run. Following our approach, a jitter constraint is imposed at the system level and then is propagated hierarchically to the circuit blocks and layout, using behavioral modeling and simulation. Experimental results are presented from working fabricated parts.

1 Introduction

The complexity of analog mixed-signal electronic systems has been increasing rapidly over the past years. Since, unlike its digital counterpart, analog circuit design is not supported by fully automatic synthesis tools, there is a great need for efficient tools and techniques to accelerate the analog design cycle. To facilitate the design of analog and mixed analog-digital circuits, we have proposed a "Top-Down, Constraint-Driven Design Methodology" [1]. The key idea of the methodology is the *hierarchical propagation* of constraints based on *behavioral modeling* and *optimization*. At each level of the design hierarchy, performance constraints are mapped onto constraints on the parameters characterizing the blocks of the subsequent level of the hierarchy. At the highest level, behavioral simulation and optimization can be used to evaluate different architectures. Once an architecture has been chosen, the process is repeated until the layout is generated or a module meeting the constraints is found in the library. Behavioral modeling and simulation allow for *early detection* of design faults and *efficient exploration* of the design space. Since models have to be estimated at high levels in the hierarchy, a *bottom-up verification* is also essential to fully characterize components, interconnects and parasitics.

Presented in this paper is the design process for a video driver system. New behavioral modeling, optimization, and layout techniques have been developed or extended from existing ones, in order to provide a full set of tools supporting the design of a class of similar mixed-signal systems. This description focuses on the critical path of the design. At the high-level synthesis phase, the frequency synthesizer phase-locked loop (PLL) behavioral models and simulation techniques are described in detail. The setup of the PLL optimization problem that performs the constraint mapping, together with the appropriate optimization algorithm are also described. Included is a jitter constraint which is set at the system level and mapped onto circuit level constraints. This is a novel approach for the design of such systems since typically jitter is measured after fabrication and, if simulation is used, it is only performed at the circuit level. Our approach can help avoid the cost of expensive design and fabrication iterations. Following the critical path of the design, the voltage-controlled oscillator (VCO) synthesis phase is depicted, with focus on the optimization approach that takes into account layout parasitics. The layout constraints generated at the circuit level are enforced during the VCO layout synthesis phase. Finally, detailed extraction of the sub-blocks and behavioral system-level simulation is used for the verification of the PLL performance.

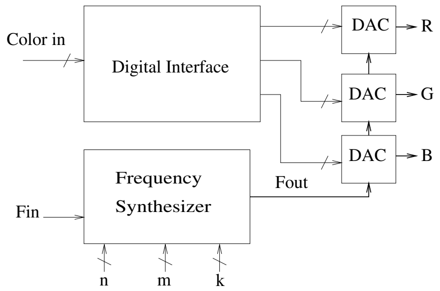


Figure 1: Display driver system diagram

2 System Description

The video driver system implemented is intended to generate the red, green, blue current signals and the synchronizing clock for video monitors in various display modes. It includes three basic subsystems: a PLL-based programmable frequency synthesizer, three D/A converters, and a digital interface file

Type	*Specification*	*Value*
Performance	Output Frequencies	25 to 135 MHz
	Timing jitter	$\leq$ 1%
	Video signal INL	$\leq$ 1 LSB
	Video signal DNL	$\leq$ 0.5 LSB
	DAC resolution	8 bits
Operation	Supply voltage	5 V
Technology	Spice models	HP CMOS34
	Design rules	SCMOS

Table 1: Video driver system specifications

register for loading the D/A converters and programming the frequency synthesizer. This system is similar to commercial display drivers except that the SRAM lookup table is not implemented. A general block diagram of the system is shown in Figure 1. The specifications for the system are given in Table 1. The synthesizer needs to generate a wide range of frequencies to support different display modes.

3 High-Level Design

The idea of hierarchical design is not new by itself; what makes this methodology valuable is providing the necessary tools and techniques for fast and efficient hierarchical mapping of the constraints. Therefore the behavioral simulation and optimization tools used will be described in detail.

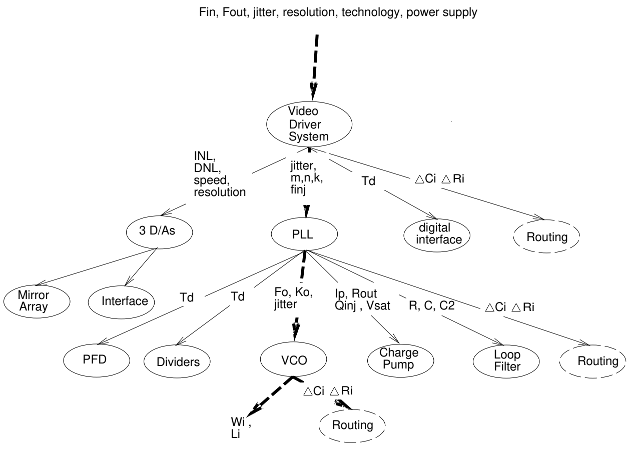


Figure 2: Video Driver System Hierarchy

The hierarchy of our design example contains two high-level decompositions (Figure 2). For the first, the constraints of Table 1 can be trivially decomposed into D/A and frequency synthesizer constraints. The D/A converter synthesis hierarchy stops after the constraints are given, since a module generator [2] is used for automatic synthesis from specifications. For the design of the file register, standard cell libraries are used. The description of the methodology will focus on the path highlighted in Figure 2. It is important to note that at each level of the hierarchy, the performance deterioration due to routing parasitics is taken into account.

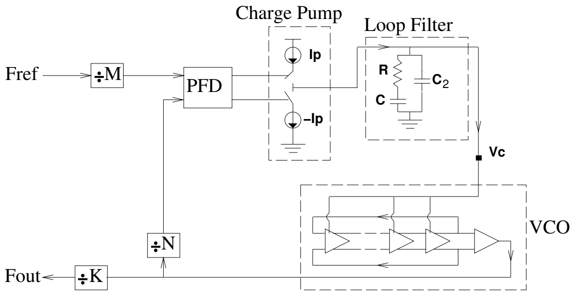


Figure 3: PLL programmable frequency synthesizer

For the PLL system design, constraints obtained from the previous level of the hierarchy will be mapped onto an architecture and component parameter constraints. In this example only one architecture is optimized. Typically though, more architectures can be evaluated using behavioral simulation.

The architecture selected is a charge-pump PLL (Figure 3) using a ring oscillator VCO and a phase-frequency detector (PFD). The main advantage of this architecture is that it does not require any external components and hence it can be easily integrated. By changing the divider values, various integer fractions of the input clock can be realized: $F_{out} = \frac{N}{M \cdot K} \cdot F_{ref}$.

PLL Behavioral Models

For the high-level mapping, a behavioral description of the PLL has to be used. It is important that the behavioral models are *implementation independent* and capture all the important second order effects determining the performance of analog circuits. A modified version of an event-driven behavioral simulator for PLLs [3] was used, including more accurate behavioral modeling of effects such as the PFD dead zone, charge-pump charge injection and mismatch, and VCO saturation and nonlinearities. The PLL is described by a set of differential equations:

$$\frac{\partial \theta_i}{\partial t} = 2\pi f_i(t) \quad (1)$$

$$\frac{\partial V_c}{\partial t} = \frac{I_{p_{eff}}}{C_2} - \frac{1}{RC_2}V_c + \frac{1}{RC_2}V_x \quad (2)$$

$$\frac{\partial V_x}{\partial t} = \frac{1}{RC}V_c - \frac{1}{RC}V_x \quad (3)$$

$$\frac{\partial \theta_j}{\partial t} = 2\pi F(V_c(t))n_d, \qquad \forall j, j = 1 \ldots n_d \quad (4)$$

where:

$$I_{p_{eff}} = ST \cdot I_p(1 + \frac{\Delta I_p}{I_p} \cdot ST) - ST\frac{\Delta V}{R_{out}} \quad (5)$$

$$F(V_c(t)) = F_0 + K_0 V_c + \ldots + K_n V_c^n, \; V_c > V_{sat} \quad (6)$$

The state variables θ_i, V_c, V_x, θ_j represent the phase of the input clock, the VCO control voltage, the voltage on capacitor C, and the phases of the n_d stages of the VCO delay stages respectively. $ST = 0, -1, 1$, depending on the state of the PFD. $F(V_c(t))$ is the instantaneous VCO frequency.

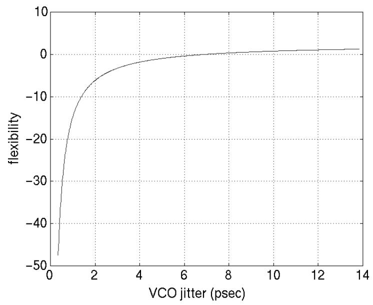


Figure 4: Jitter Flexibility Function

The PFD is modeled as a state-transition table. The state transition events happen at the zero-crossing of the VCO output, i.e. $\theta(t_0) = n\pi$. An iterative integration method is used to compute the exact transition times, so that numerical noise is minimized.

Even though many effects such as power supply and substrate coupling can contribute to the overall timing jitter, the fundamental performance limit is due to the devices' intrinsic thermal noise. If careful design techniques are used, such as differential architectures, separate power supplies and on-chip decoupling capacitors, most coupling effects can be reduced so that PLL timing jitter can be attributed mainly to thermal noise, which is modeled as a white Gaussian random process. The overall jitter is then predicted by adding random noise at the time of each VCO transition and subsequently processing the resulting waveform [3].

PLL High-Level Optimization

Since the behavioral description does not depend on the low-level implementation, we choose the high-level parameters by optimizing for maximum *design flexibility* [1]. Flexibility is a heuristic measure of the easiness to meet a set of design specifications. Typically, parabolic and hyperbolic functions are used. The flexibility function for parameter $\Delta\tau_{VCOrms}$ is shown in Figure 4. The criterion used to build the flexibility functions was attributing $flex(x) = -10$ for a parameter value considered "hard", and $flex(x) = 0$ for a parameter value considered "easy" to obtain. Those parameters were heuristically adjusted. By using flexibility functions it is possible to consider design trade-offs at the system level in a systematic way, without knowing the details of the implementation. This significantly accelerates the design process.

The performance constraints of the PLL are, stability in the frequencies of operation, and timing jitter. Stability is checked for the worst case configuration by imposing a maximum acquisition time. Jitter is also checked at the worst jitter accumulation configuration. To ensure tolerance to parameter variations, that can be as high as 30% of the nominal value, an additional *phase margin* constraint is added. The optimization problem can therefore be expressed as:

$$max \quad \sum_{i=1}^{n} flex_i(x_i) \tag{7}$$

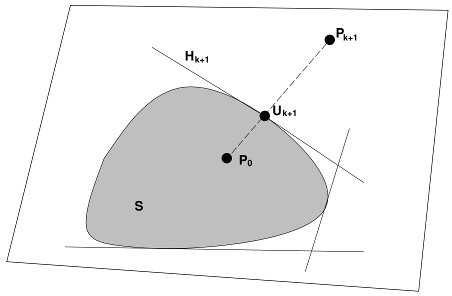


Figure 5: Supporting Hyperplane Method

$$s.t. \quad \angle \left[2\pi\ (\mathrm{C}+\mathrm{C}_2)\ \mathrm{s}^2 \frac{1+\mathrm{s\,R\,C}}{1+\mathrm{sR}\frac{\mathrm{C\,C_2}}{\mathrm{C+C_2}}} \right]_{\mathrm{f_u}} \quad \geq \quad -135^o \tag{8}$$

$$\Delta\tau_{rms}\,|_{f_{max}} \quad \leq \quad 50\,ps \tag{9}$$

$$T_{acq} \quad \leq \quad T_{max} \tag{10}$$

where $f_{max} = 140\ MHz$, f_u is the unity gain frequency, and n is the number of parameters used in the optimization: $K_o, \Delta\tau_{VCO}, I_p, R, C, C_2$.

Most nonlinear optimization algorithms require accurate computation of the first and/or second order derivatives for convergence reasons. This may be difficult to obtain when simulators are used to calculate the constraints. Furthermore, the gradients of the constraint function are often not defined outside the feasible region. This is the case of the PLL, where timing jitter and acquisition time cannot be defined when the system is unstable.

A quite efficient method to address such problems is the *supporting hyperplane* method. The algorithm operates as follows: after an initial feasible point is given, an unconstrained optimization is performed. Then, all nonlinear constraints are checked and if the solution point $\mathbf{P_{k+1}}$ is feasible, the algorithm stops and the solution is a global minimum. If a constraint g_i is violated, then the point $\mathbf{u_{k+1}}$ is found on the line joining the initial feasible point $\mathbf{P_0}$ and the last solution $\mathbf{P_{k+1}}$, that lies on the boundary of the feasible region S. Then a linear constraint is added such as: $\nabla g_j(\mathbf{u_{k+1}})(\mathbf{x} - \mathbf{u_{k+1}}) \leq 0$. Consequently, the linearized constrained optimization problem is solved again. This process is repeated until a global minimum is found that satisfies the nonlinear constraints. The algorithm is depicted graphically in Figure 5. In order to guarantee convergence to a global optimum, the feasible space must be convex.

In this algorithm, derivatives are only needed in the feasible space, where the constraint functions are well defined. In the case of the PLL a great problem is eliminated, since the jitter constraint is not defined when the system is unstable. However, the convexity requirement is a significant drawback since it is hard to guarantee in most circuit design problems. Even though it worked in the specific PLL case, the algorithm could fail in more complicated optimization problems with more variables.

The algorithm was implemented in C++. Behavioral simulation was used to compute the jitter and stability constraints. The initial feasible point, found using behavioral simulation,

Parameters	*Final*	*Initial*
K_0 (MHz/V)	40	50
$\Delta\tau_{VCO_{rms}}$ (ps)	3.33	1.03
I_p (μA)	15.8	5
R ($K\Omega$)	200.5	220
C (pF)	57.8	220
C_1 (pF)	5	5
Constraints	*Final*	*Initial*
$\Delta\tau_{rms} \leq 50$ ps (ps)	50.42	45
Phase margin $\geq 45^o (^o)$	43.6	60
Flexibility	2.79	-48.9
CPU Time (sec)	7606.1	
Iterations	11	

Table 2: Optimization results

was externally provided to the optimizer. The phase margin constraint was computed first to save CPU time. Since the jitter constraint is the result of a Monte-Carlo simulation, the gradients computed can be quite inaccurate. An iterative method was used to define the step for the finite differences. A large step within the feasible region was initially used, which was reduced until the value of the derivative became noisy. If a solution to the linear optimization problem could not be found, the point at which the derivative was computed was moved more "within" the feasible region. The possible loss of overall optimality is of little concern, since a heuristic objective is used. The results of the high-level optimization are summarized in Table 2. The tolerance of the optimization result to worst-case parameter variations was verified using behavioral simulation.

4 Low-Level Design

Following the methodology, the high-level parameters become *performance constraints* for the low-level building blocks and are mapped onto a sized architecture of transistors and layout parasitics. A standard dead zone-free PFD was automatically synthesized from high-level description using digital synthesis tools. For the charge-pump, a design similar to the one described in [4] was used.

For the VCO, a ring oscillator VCO topology using differential cells with CMOS loads in triode region [5, 4] was selected in order reduce the effect of power supply and substrate coupling. The oscillator consists of eight cells and its output is converted to full CMOS swing via a level-restoring circuit. A modified version of the cell topology described in [5] was used. The topology of the cell with the bias circuit is shown in Figure 6.

Optimization Taking into Account Parasitics

Following the proposed methodology, the performance constraints for the VCO must be mapped onto component values.

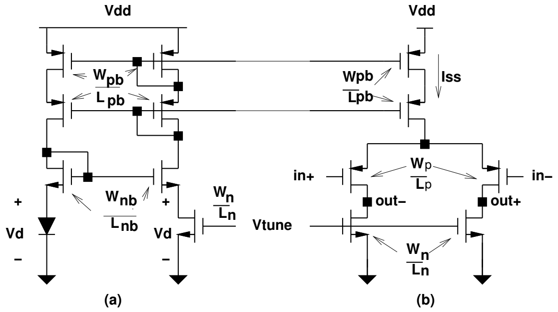


Figure 6: VCO delay cell and bias circuit

To ensure that the performance constraints are met after the layout is done, it is critical that layout parasitics are taken into account during the optimization phase.

Let **P** a performance vector, **C** the parasitics vector and $\mathbf{P_{max}}$ the corresponding maximum allowed performance degradation due to those parasitics. Assuming a linear model around the nominal performance and small parasitics, the performance degradation ΔP_i can be given by:

$$\Delta P_i = \left[\mathbf{S_C^{P_i}}\right]^T \cdot \Delta\mathbf{C} \tag{11}$$

$\mathbf{S_C^{P_i}}$ is the sensitivity vector of performance P_i with respect to the parasitics' vector **C** and Δ**C** is the deviation from the nominal estimate of the parasitics. Given a bound $\Delta\mathbf{C_{max}}$ on the maximum allowed deviations from the nominal estimate of the parasitics, we can force the optimization result to have a reduced sensitivity to parasitics by imposing a constraint on the maximum performance deterioration allowed.

The nominal estimate of the parasitics and the maximum allowed deviation are subsequently used to compute constraints for the VCO layout generator. A maximum deviation of 50% from a nominal estimate of 15 pF for the parasitics at the outputs of the differential gates were used. For the optimization, only the critical device sizes, W_n, L_n, W_p, L_p were used as parameters. The overall optimization problem for the VCO can be expressed as:

$$min \;\; Power(VCO) \tag{12}$$

$$s.t. \;\; F_{maxmin} \leq F_{maxVCO} \leq F_{maxmax} \tag{13}$$

$$F_{minmin} \leq F_{minVCO} \leq F_{minmax} \tag{14}$$

$$\Delta\tau_{VCOrms} \leq \Delta\tau_{max} \tag{15}$$

$$\left[\mathbf{S_C^P}\right]^T \cdot \Delta\mathbf{C_{max}} \leq \Delta\mathbf{P}_{max} \tag{16}$$

The optimization problem was again solved using the supporting hyperplane algorithm with the initial feasible point provided externally. All constraints were evaluated using SPICE simulations except for the timing jitter constraint that was evaluated using equations [6]. The sensitivities were evaluated using finite differences. The sizes obtained were $W_n = 2.6\,\mu m, L_n = 4\,\mu m, W_p = 36\,\mu m, L_p = 1\,\mu m.$

begin{
  for-each(P_j)
    for each(R_i, C_i) calculate($\frac{\partial P_j}{\partial C_i}$, $\frac{\partial P_j}{\partial R_i}$);
  do {
  calculate($R_{i\,max}$, $C_{i\,max}$); /* quadratic optimization*/
  for-each(i) {
  set $W_i = W_{i\,min}$ and $L_i = L_{i\,min} \implies C_i = C_0\, W_{min}\, L_{min}$;
    do {
      evaluate $R_i = \rho\, \frac{W_i}{L_i}$;
      if $(R_i < R_{i\,max})$ **then** exit;
      else $W_i = W_i + \Delta W$;
    } **while** $(C_i < C_{i\,max})$;
  } **while** $((C_i > C_{i\,max})$ or $(R_i > R_{i\,max}))$;
}**end**

Figure 7: Layout Generation Algorithm

5 Physical Design

The constraints set in the optimization problem of Equations 12 - 16 were used in the layout generation. Moreover, constraints for all other parasitics were generated using the constraint generation techniques described in [7]. The sensitivities of every performance parameter with respect to every parasitic resistance and capacitance were calculated automatically using finite differences and then, given a maximum allowable performance deviation, bounds were imposed on every parasitic using quadratic optimization maximizing layout flexibility.

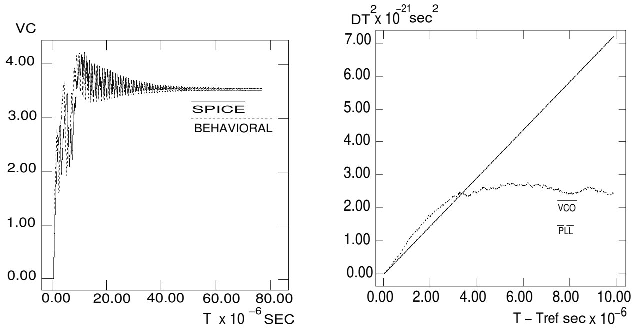
A parametric layout generator was written for the specific VCO topology. It uses a fixed floor-plan and takes as parameters the number of delay cells, the device sizes and the parasitic constraints. Additional parasitic constraints were generated for the parasitics that were not accounted for in the circuit optimization. The algorithm for the layout generation is shown in Figure 7. ΔW is the minimum increment allowed by the process design rules, P_j is the performance j and i is the number of the parametric wires.

The final layout for the video driver system was synthesized using automatic routing tools. Different analog and digital supplies were used and special supplies were provided for the VCO in order to avoid as much as possible supply-coupled noise which can contribute to timing jitter.

6 Bottom-Up Verification

The value of behavioral modeling and simulation is apparent in the verification phase of the PLL, which is an inherently "stiff" system, often causing a full circuit simulation to be impossible or unrealistic. Following the hierarchical verification approach, first the performance parameters of the PLL building blocks were extracted using SPICE. The VCO timing jitter was extracted using the non-Monte Carlo, nonlinear noise simulator described in [8]. Then, behavioral simulation was used to verify the performance of the whole system.

In Figure 8(a), the result of a flat circuit simulation is compared to the result of the behavioral simulation. The waveform simulated is the control voltage of the VCO when the PLL is in acquisition mode and is done to detect stability in the worst case divide ratio. The waveforms from both simulations are almost identical. The behavioral simulation completed in 560 CPU seconds, while the full circuit simulation took 20 CPU hours (using macro-models for the dividers). Both simulations were performed in a *DEC Alpha-Server 2100 5/250* with 256 Mb of memory and 4 CPU's. Figure 8(b) shows the result of a behavioral simulation for the timing jitter using the extracted parameters for $F_{out} = 100\ MHz$. The plot shows the square of the PLL and VCO rms timing jitter as a function of the distance from the reference transition. As expected, the open loop VCO jitter accumulates linearly to infinity, since there is no correction from the PLL loop while the PLL jitter converges to a final value. The projected performance is based only on the calculation of the thermal jitter of the VCO, which sets the fundamental performance bound. Still, the performance of the actual system is expected to be close to the one predicted since care has been taken to reduce as much as possible all other jitter sources.



(a) (b)

Figure 8: PLL Verification (a) acquisition (b) jitter

7 Experimental Results

The chip was fabricated on a MOSIS HP $1.0\mu m$ technology. A die photo is shown in Figure 9. The 17,000 transistor system occupies an area of 3.4 mm x 3.9 mm= 13.26 mm^2. A

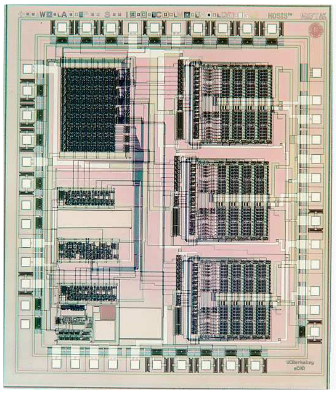


Figure 9: Video Driver System Die Photo

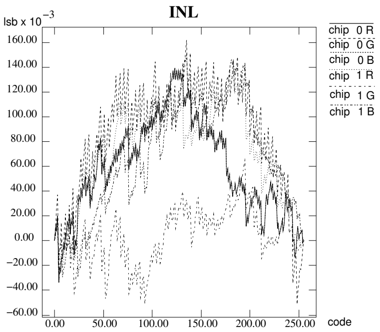


Figure 10: INL Measurement Results

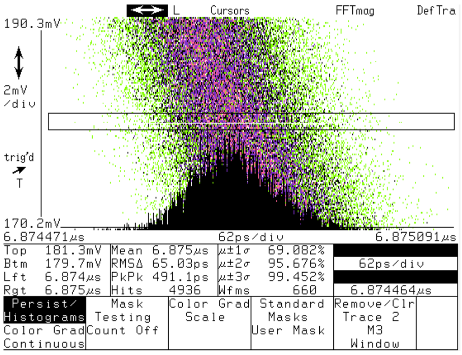


Figure 11: Jitter Histogram

printed circuit board was designed and manufactured in order to measure the performance of the chip. Experimental results show that the D/A INL and DNL performance is 0.16 LSB and 0.05 LSB respectively and that the settling speed requirements are also met (T_{set} = 6 nsec). Figure 10 shows experimental INL data from six D/A converters as a function of the input code.

The PLL frequency generator meets the specifications for generating frequencies from *25 MHz to 130 MHz*. Figure 12 shows the output waveform at 130 MHz. A small deviation from the expected speed in the upper edge of the specifications is due to an error in the parameters file used in the synthesis phase. Detailed timing jitter measurements were done using a *Tektronix 11801B* high bandwidth digitizing oscilloscope with the same waveform feeding the signal and the trigger inputs. Figure 11 shows an output waveform at 100 MHz and the corresponding jitter histogram at a transition edge 7 μs from the reference, so that the accumulated jitter converges to its final value. The rms jitter at 100 MHz is *65 ps* (0.65 %), which is close to the specifications. The results are in agreement with predictions within 30 % for the worst case chip. Component process variations affecting the PLL bandwidth, simplified noise models for the devices, power supply and substrate coupling can cause the measured value to deviate from our predicted value. Also reflections and coupling from the testing board can significantly affect the measurements. For this reason, the agreement between results and predictions is quite satisfactory.

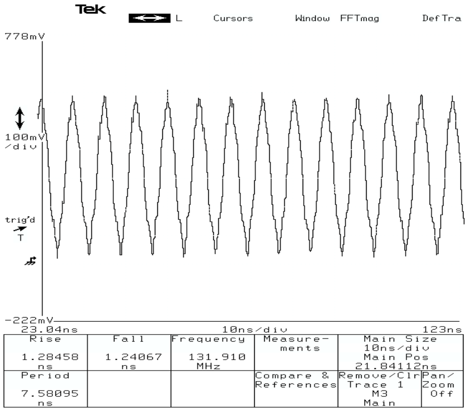


Figure 12: Frequency Synthesizer Output

8 Conclusions

A complete design flow for a video driver system has been presented, based on the top-down, constraint-driven paradigm. Experimental results verify the validity of the design methodology. Fundamental to the approach was the use of behavioral simulation and optimization for hierarchical constraint propagation. Combined with the tools used, this methodology can have a significant impact on the design of similar systems by reducing over-design, design times and costly fabrication iterations.

Acknowledgments

This research was supported by SRC (96-DC-324).

References

[1] H. Chang, A. Sangiovanni-Vincentelli, F. Balarin, E. Charbon, U. Choudhury, G. Jusuf, E. Liu, E. Malavasi, R. Neff and P. Gray, "A Top-down, Constraint-Driven Design Methodology for Analog Integrated Circuits", in *Proc. IEEE CICC*, pp. 841–846, May 1992.

[2] R. Neff, P. Gray and A. Sangiovanni-Vincentelli, "A Module Generator for High Speed CMOS Current Output Digital/Analog Converters", in *Proc. IEEE CICC*, pp. 481–484, May 1995.

[3] A. Demir, E. Liu, A. Sangiovanni-Vincentelli and I. Vassiliou, "Behavioral Simulation Techniques for Phase/Delay-Locked Systems", in *Proc. IEEE CICC*, pp. 453–456, May 1994.

[4] I. A. Young, J. K. Greason and K. L. Wong, "A PLL Clock Generator with 5 to 110 MHz of Lock Range for Microprocessors", *JSSC*, vol. 27, n. 11, pp. 1599–1607, November 1992.

[5] D. Reynolds, "A 320MHz CMOS Triple 8b DAC with On-Chip PLL and Hardware Cursor", in *Proc. IEEE International Solid-State Circuits Conference*, pp. 50–51, February 1994.

[6] T. C. Weigandt, B. Kim and P. R. Gray, "Analysis of Timing Jitter in CMOS Ring Oscillators", in *Proc. IEEE Int. Symposium on Circuits and Systems*, May 1994.

[7] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint Generation for Routing Analog Circuits", in *Proc. IEEE/ACM DAC*, pp. 561–566, June 1990.

[8] A. Demir, E. Liu and A. Sangiovanni-Vincentelli, "Time-Domain non-Monte Carlo Noise Simulation for Nonlinear Dynamic Circuits with Arbitrary Excitations", in *Proc. IEEE ICCAD*, pp. 598–603, November 1994.