Modeling and Verification of High-Speed Wired Links with Verilog-AMS

Ming-ta Hsieh
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455, USA
mthsieh@umn.edu

Gerald E. Sobelman
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455, USA
sobelman@umn.edu

***Abstract*—Behavioral modeling with virtual built-in self-test verification of high-speed wired link designs is described in this paper. Our procedure is based on principles of top-down mixed-signal design combined with a behavioral description language and mixed-mode simulations. The use of Verilog-AMS is applied not only to circuit modeling but also for representing noise on the input signal. This approach provides system-level jitter tolerance estimation, circuit critical path search and overall design verification. Coding examples and simulation results are included.**

I. Introduction

The idea of top-down mixed-signal circuit design using a behavioral description for system-level modeling has been a topic of research interest [1-3]. Typically, system architecture designers derive the required system or circuit specification for certain targeted applications. On the other hand, circuit designers would search for a type of circuit which is the best fit for the required specifications with minimum design cost and time and having superior performance. Before going ahead with the detailed transistor-level design, behavioral models are usually used to verify the design at the system level in order to determine which type of circuit has the best performance. During the design process, transistor-level circuits may be mixed with behavioral models for purposes of performance evaluation. After completing each transistor-level circuit design, the corresponded behavioral circuit for each module is tweaked to match to its transistor-level circuit performance. By the end of the design process, a more thorough system-level interconnection check and performance verification are executed. This final system–level verification is done based on either mixed-mode or fully behavioral simulations. The mixed-mode simulation includes both transistor and behavioral modules which have their critical path delays based on transistor-level extraction. Of course, an accurate behavioral model is needed in order to provide useful information for the total system-level verification. A complete transistor-level system performance evaluation is not practical due to the enormous simulation time that would be required [4]. Using behavioral modeling lets designers easily explore different system-level architectures at the early design stages and thereby rapidly perform system-level verification, Thus, including behavioral models in the overall design process reduces time to market and helps to ensure first time correct silicon.

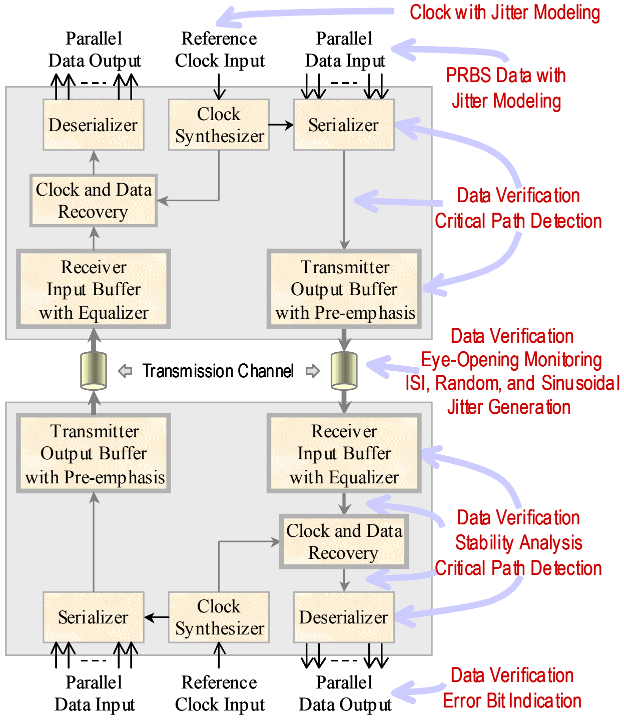

Clock with Jitter Modeling
Parallel Data Output
Reference Clock Input
Parallel Data Input
PRBS Data with Jitter Modeling
Deserializer
Clock Synthesizer
Serializer
Clock and Data Recovery
Data Verification Critical Path Detection
Receiver Input Buffer with Equalizer
Transmitter Output Buffer with Pre-emphasis
Data Verification Eye-Opening Monitoring ISI, Random, and Sinusoidal Jitter Generation
Transmission Channel
Transmitter Output Buffer with Pre-emphasis
Receiver Input Buffer with Equalizer
Clock and Data Recovery
Data Verification Stability Analysis Critical Path Detection
Serializer
Clock Synthesizer
Deserializer
Parallel Data Input
Reference Clock Input
Parallel Data Output
Data Verification Error Bit Indication

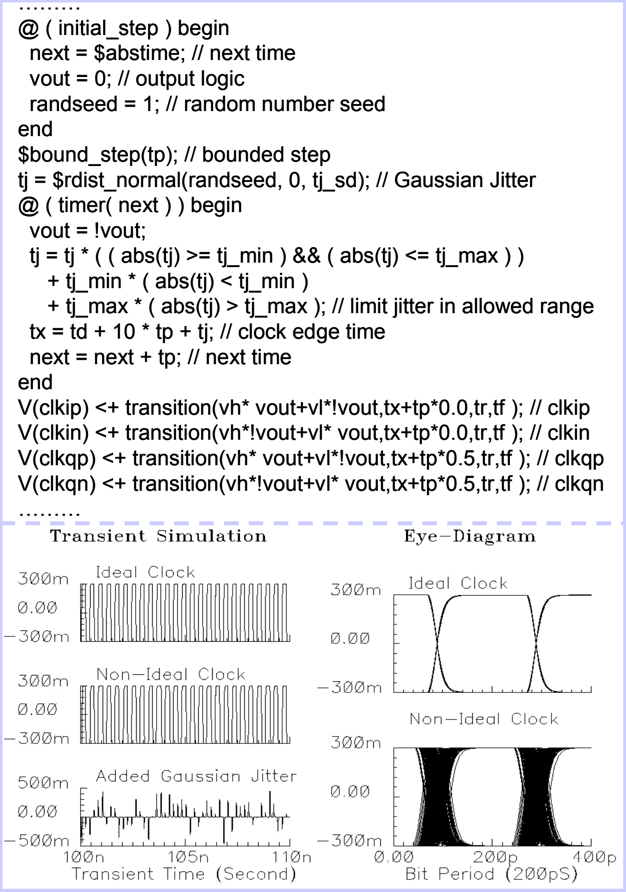
Figure 1. Application of behavioral modeling to high-speed link design.

Figure 1 shows an example of the use of behavioral circuit models applied to high-speed wired link designs. The behavioral models may also be used to generate inputs such as clocks or PRBS data associated with jitter, to search for internal critical paths, to verify correctness of data transmission with virtual build-in-self-test and so on. The ultimate goal is not only to use the behavioral description to model circuits for system-level verification but also to identify the weak points in a system for subsequent analysis.

II. INPUT SIGNAL MODELING

In the real world, inputs are never ideal and are always associated with some kind of noise. In the time domain, the noise on clocks or input data is known as jitter. This section demonstrates the use of behavioral modeling to generate these non-ideal signals.

A. *Clock with Jitter Modeling*

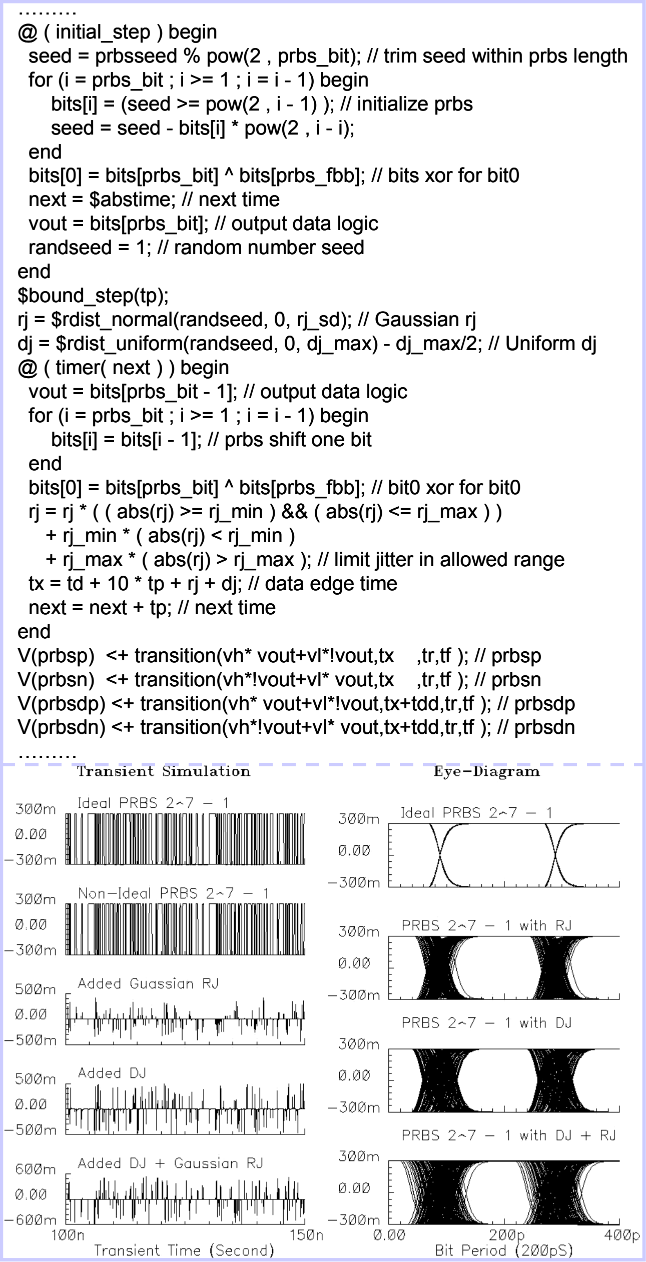


```
………
@ ( initial_step ) begin
  next = $abstime; // next time
  vout = 0; // output logic
  randseed = 1; // random number seed
end
$bound_step(tp); // bounded step
tj = $rdist_normal(randseed, 0, tj_sd); // Gaussian Jitter
@ ( timer( next ) ) begin
  vout = !vout;
  tj = tj * ( ( abs(tj) >= tj_min ) && ( abs(tj) <= tj_max ) )
     + tj_min * ( abs(tj) < tj_min )
     + tj_max * ( abs(tj) > tj_max ); // limit jitter in allowed range
  tx = td + 10 * tp + tj; // clock edge time
  next = next + tp; // next time
end
V(clkip) <+ transition(vh* vout+vl*!vout,tx+tp*0.0,tr,tf ); // clkip
V(clkin) <+ transition(vh*!vout+vl* vout,tx+tp*0.0,tr,tf ); // clkin
V(clkqp) <+ transition(vh* vout+vl*!vout,tx+tp*0.5,tr,tf ); // clkqp
V(clkqn) <+ transition(vh*!vout+vl* vout,tx+tp*0.5,tr,tf ); // clkqn
………
```

Figure 2. Example of modeling a clock with random jitter.

The clock synthesizer module as shown in Figure 1 often requires a reference clock input. This clock input normally contains a certain amount of random jitter (RJ). [5] This RJ information is needed when performing system-level versification or clock-synthesizer design in a high-speed wired link. For the other individual module designs such as the serializer and clock data recovery (CDR) of Figure 1, the major concern is the impact of jitter from the output clock of the synthesizer. Reference [6] provides a detailed analysis of this impact in high-speed serial links. An example of a clock with RJ modeling is shown in Figure 2. RJ typically has a Gaussian distribution. The magnitude of RJ is a function of the desired bit error rate (BER) for the system. [7] The expected jitter magnitude and time interval for the RJ distribution are taken as inputs to generate the RJ in this example.

B. *Data with Jitter Modeling*



```
………
@ ( initial_step ) begin
  seed = prbsseed % pow(2 , prbs_bit); // trim seed within prbs length
  for (i = prbs_bit ; i >= 1 ; i = i - 1) begin
     bits[i] = (seed >= pow(2 , i - 1) ); // initialize prbs
     seed = seed - bits[i] * pow(2 , i - i);
  end
  bits[0] = bits[prbs_bit] ^ bits[prbs_fbb]; // bits xor for bit0
  next = $abstime; // next time
  vout = bits[prbs_bit]; // output data logic
  randseed = 1; // random number seed
end
$bound_step(tp);
rj = $rdist_normal(randseed, 0, rj_sd); // Gaussian rj
dj = $rdist_uniform(randseed, 0, dj_max) - dj_max/2; // Uniform dj
@ ( timer( next ) ) begin
  vout = bits[prbs_bit - 1]; // output data logic
  for (i = prbs_bit ; i >= 1 ; i = i - 1) begin
     bits[i] = bits[i - 1]; // prbs shift one bit
  end
  bits[0] = bits[prbs_bit] ^ bits[prbs_fbb]; // bit0 xor for bit0
  rj = rj * ( ( abs(rj) >= rj_min ) && ( abs(rj) <= rj_max ) )
     + rj_min * ( abs(rj) < rj_min )
     + rj_max * ( abs(rj) > rj_max ); // limit jitter in allowed range
  tx = td + 10 * tp + rj + dj; // data edge time
  next = next + tp; // next time
end
V(prbsp)  <+ transition(vh* vout+vl*!vout,tx    ,tr,tf ); // prbsp
V(prbsn)  <+ transition(vh*!vout+vl* vout,tx    ,tr,tf ); // prbsn
V(prbsdp) <+ transition(vh* vout+vl*!vout,tx+tdd,tr,tf ); // prbsdp
V(prbsdn) <+ transition(vh*!vout+vl* vout,tx+tdd,tr,tf ); // prbsdn
………
```

Figure 3. Example of modeling data having ISI and random jitter.

A bandwidth-limited transmission channel causes intersymbol interference (ISI) on data, while an imperfect clock injects RJ into the data stream. Other jitter associated with the data are cross-talk (XT) and duty-cycle distortion (DCD). XT can be treated as a superposition of noise and signal. DCD is a skewing of the pulse width. Both ISI and DCD are part of deterministic jitter (DJ). A discussion of jitter associated with data can be found in Ref. [8] and behavioral modeling of jitter is discussed in Ref. [9]. An example of data having RJ and DJ is shown in Figure 3, which presents a pseudorandom bit sequence (PRBS) data with a jitter generator. For the deserializer and the transmit driver designs, RJ is typically the only type included. For the receiver input and CDR design, however, both RJ and DJ should be considered.

III. SYSTEM AND CIRCUIT MODELING

System and circuit behavioral modeling allow designers to explore different system architectures or circuit topologies and make a quick feasibility analysis in the early stages of design [10]. Moreover, behavioral modeling is often used to prove the correctness of the fundamental theory and to estimate the overall performance of the system before starting the transistor-level design [3]. The goal is to provide an efficient system-level or hierarchical mixed-mode design verification platform.

A. System-Level Modeling

Behavioral modeling at the system-level often starts from a functional description in order to provide an intuitive feel for the signal flow from input to output as well as the overall system functionality. For example, a high-speed serial link starts with low-frequency parallel data sampled by a synthesized clock and serialized into a stream of data. The serialized data is then pre-emphasized and transmitted to the receiver through a bandwidth-limited channel. The receiver needs to equalize the received signal in order to compensate the high frequency spectrum loss before extracting the clock and data from the CDR loop. Next, the extracted data is sampled by the recovered clock and deserialized back into a parallel set of signals. Once a preliminary behavioral model has demonstrated the expected functional operation, then non-ideal components for jitter described in Section II are included to estimate jitter tolerance.

B. Circuit Level Modeling

Individual circuit modules below the system-level should also be modeled using behavioral descriptions. Such a circuit model could be used as an input signal provider or an output signal checker when running a transistor-level performance simulation for a specific circuit block. On the other hand, the circuit model may also be used for system-level mixed-mode or purely behavioral design verification.

IV. DESIGN VERIFICATION

The design verification process may be divided into module and system levels of performance evaluation It can be classified as being either full behavioral, mixed-mode or a complete transistor-level verification. Based on principles of top-down mixed-signal design, the design description proceeds from top to bottom (i.e., system-level to circuit level), and from purely behavioral to a purely physical level. On the other hand, the verification process may run from bottom to top and from physical to behavioral descriptions. Such a performance evaluation process is called Bottom-Up Verification [3].

A. Test Bench

We refer to a test bench based on behavioral modeling as virtual built-in self-test (VBIST). The primary focus of VBIST is to verify the designed physical circuit not only under different process, voltage and temperature (PVT) conditions, but also under extreme values of the input signals. For example, the receiver input stage which includes an input buffer, DC-restore and equalizer could have different output responses based on an AC- or DC-coupled connection, the input common-mode voltage, the degree of duty-cycle distortion and the magnitude of DJ and RJ. If the input buffer gain and the equalizer frequency compensation are also variable, there will be on the order of 10 variations to consider during the analysis. The goal of VBIST is to create a test vector generator based on behavioral models that can accommodate all of these types of variations.

B. Critical Path Detector

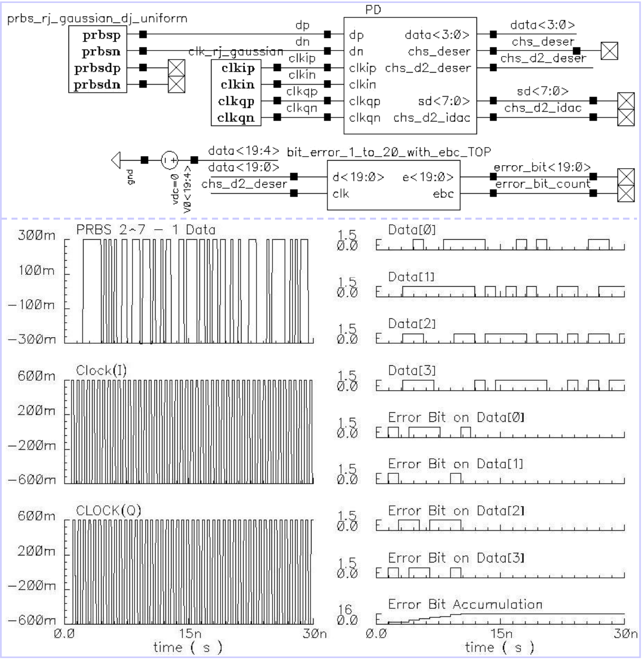


Figure 4. Phase detector modeling and simulation.

The primary purpose of critical path detection is to identify the components which are most likely to degrade the overall performance in a designed circuit or system. For example, the phase detector (PD) in the CDR block samples the input signal at the receiver input stage and often demultiplexes the sampled stream into two (or more) parallel output signals [11]. One of the output signals may have a higher probability of bit errors than the other signal. A critical path detector must be able to identify the weaker path and the location of the failing component. An example of such a simulation is shown in Figure 4. Of course, the same concepts can also be applied to serializer and deserializer design verification.

C. *System Level Verification*

```
.........
@ ( cross( V(clk) - vt_clk , 1 ) ) begin
  ct = $abstime; // get current time
  dx[ 0] = ( V(d00) > vt_d ); // transfer input data into dx in logic level
  .........
  dx[19] = ( V(d19) > vt_d ); // from bit 0 to bit 19
  err = 0; // set sum of error bits to be zero for 1st set comparison
  for ( i = 0 ; i < 30 - input_bits ; i = i + 1 ) begin
      p[i] = p[i + input_bits];
      d[i] = d[i + input_bits];
      e[i] = ( d[i] != p[i] );
      err = err + e[i];
  end
  for ( i = 30 - input_bits ; i < 30 ; i = i + 1 ) begin
      p[i] = ( p[i - prbs_bit] + p[i - prbs_fbb] ) % 2;
      d[i] = dx[i - 30 + input_bits];
      e[i] = ( d[i] != p[i] );
      err = err + e[i];
  end
end
if (err > 0) begin
    errx = 0; // set sum of error bits to be zero for 2nd set comparison
    for ( i = 0 ; i < prbs_bit ; i = i + 1 ) begin
        px[i] = d[i];
        ex[i] = ( d[i] != px[i] );
        errx = errx + ex[i];
    end
    for ( i = prbs_bit ; i < 30 ; i = i + 1 ) begin
        px[i] = ( px[i - prbs_bit] + px[i - prbs_fbb] ) % 2;
        ex[i] = ( d[i] != px[i] );
        errx = errx + ex[i];
    end
    if (errx == 0) begin
        for ( i = 0 ; i < 30 ; i = i + 1 ) begin
            p[i] = px[i];
            e[i] = ex[i];
        end
    end
    err = 0; // set sum of error bits to be zero for initial set comparison
    if ( ( ct - st ) >= ts_ebc ) begin
       for ( i = 0 ; i < input_bits ; i = i + 1 ) begin
           eb_cnt = eb_cnt + e[ 30 - input_bits +  i ]; // counting error bits
       end
    end
end
V(e00) <+ transition(voh*e[30-input_bits+ 0]
              +vol*!e[30-input_bits+ 0],td_e,tr_e,tr_e);
.........
V(e19) <+ transition(voh*e[30-input_bits+19]
              +vol*!e[30-input_bits+19],td_e,tr_e,tr_e);
V(ebc) <+ transition(eb_cnt, td_e, tr_e, tr_e);
.........
```

Figure 5. Verification model of the bit error checker.

Overall system-level performance is evaluated through a top-level verification. The first step of system-level verification is to check the interconnection of components and the overall functionality. This step is often done at the transistor-level in order to make sure that the whole system is working together and the physical layout has been done correctly. However, a detailed system-level performance evaluation including jitter generation, jitter tolerance and bit error rate (BER) analysis is conducted using mixed-mode or purely behavioral simulations [4]. Figure 5 shows an example of the bit error checker used in system-level verification.

V. CONCLUSIONS

In this paper, we have demonstrated the use of behavioral modeling with Verilog-AMS for the design and verification of high-speed wired links. In the past, behavioral modeling was primarily used by a system architect in order to estimate overall system performance. This paper extends the use of behavioral models to handle non-ideal input signal generation and associated design verification issues in order to account for these important real-world effects. The ultimate goal of such behavioral modeling is to better ensure first-pass functioning silicon. Furthermore, behavioral models may be more easily reused in different design projects since their construction is not closely dependent on the details of a given process technology.

REFERENCES

[1] R. Sommer et. al., “From System Specification to Layout: Seamless Top-Down Design Methods for Analog and Mixed-Signal Applications,” IEEE Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition

[2] P. Muller, A. Tajalli, M. Atarodi, and Y. Leblebici, “Top-down design of a low-power multi-channel 2.5-Gbit/s/channel gated oscillator clock-recovery circuit,” IEEE Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition

[3] K. Kundert, “Principles of Top-Down Mixed Signal Design,” The Designer’s Guide Community http://www.designers-guide.org, 2003

[4] M. Wang, H. Maramis, D. Telian, and K. Chung, “ New Techniques for Designing and Analyzing Multi-GigaHertz Serial Links,” DesignCon 2005.

[5] C. Skach, “Signals predict Serdes jitter behavior,” EE Times Asia, http:// www.eetasia.com, May 16, 2005.

[6] P.K. Hanumolu, B. Casper, R. Mooney, G. Wei, and U. Moon, “Analysis of PLL clock jitter in high-speed serial links,” IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol.50, No.11, November 2003, pp. 879-886.

[7] M.A. Kossel and M.L. Schmatz, “Jitter Measurements of High-Speed Serial Links,” IEEE Design and Test of Computers, Volume 21, Issue 6, pp. 536 – 543, No-vember 2004

[8] Lattice Semiconductor Corporation “Technical Note TN1084: SERDES Jitter,” Lattice Semiconductor Corpo-ration, March 2, 2005.

[9] M. Hsieh and G. Sobelman, “Accurate Prediction of Jitter Tolerance in High-Speed Serial Links,” to appear in *International SoC Design Conference*, 2005.

[10] B.Antao, F. El-Turky, and R. Leonowich, “Mixed-mode simulation of phase-locked loops,” IEEE 1993 Custom Integrated Circuits Conference, 9-12 May 1993, pp. 8.4.1 - 8.4.4

[11] A. Rezayee, and K. Martin, "A 9-16Gb/s Clock and Data Recovery Circuit with Three state Phase De-tector and Dual-Path Loop Architecture," Proceedings of the 2003 European Solid-State Circuits Conference, Estoril, Portugal, pp. 683-686.