Timing Characterization of a Digital Cell Library

Sanjeev Jindal

Matt Silverman
Objective

- To create Verilog-AMS test benches to characterize the OSU Digital Cell Library
- To automatically characterize the cells using a script
Motivation

- To allow easier and quicker ways to do timing characterizations of cells
- Doing the characterization with the cadence GUI can be time consuming
- More useful in characterizing analog cells due to the number of parameters
Cells in Library

Test Benches Structure

- The test benches contain:
  - A stimulus block which applies different voltage values to the inputs of the gates
  - A results extractor which looks at the outputs and inputs of the gates and calculates the timing information
  - The tested cell
  - A load (nand2x1)
Schematic for Buffer
Verilog AMS Stimulus Block

```
`include "constants.vams"
`include "disciplines.vams"
`timescale 1ns/1ps

module bufx1stim(vouta,vdd,vss,vmid);
  output vouta,vdd,vss,vmid;
  electrical vouta,vdd,vss,vmid;
  reg a;
  parameter real VoltageRail=3.3;
  initial begin
    a=0;
    #10 a=0;
    #10 a=1;
    #10 a=0;
  end

  analog begin
    V(vdd,vmid) <+ VoltageRail/2;
    V(vmid,vss) <+ VoltageRail/2;
    V(vouta,vss) <+ a*VoltageRail;
  end

endmodule
```
Results Extractor

@((cross(V(vina,vmid)))) begin
    starttimea=$abstime;
    risetimea=0;
    falltimea=0;
    ns_a=V(vina);
    end

@((cross(V(vouty,vmid)))) begin
    starttimey=$abstime;
    ns_y=V(vouty);
    if (ns_a > cs_a && ns_y > cs_y) begin
        risetimea=starttimey-starttimea;
        $strobe("Y rise time to A rise time is %f", risetimea/1n);
    end
    if (ns_a > cs_a && ns_y < cs_y) begin
        risetimea=starttimey-starttimea;
        $strobe("Y fall time to A rise time is %f", risetimea/1n);
    end
    if (ns_a < cs_a && ns_y > cs_y) begin
        falltimea=starttimey-starttimea;
        $strobe("Y rise time to A fall time is %f", falltimea/1n);
    end
    if (ns_a < cs_a && ns_y < cs_y) begin
        falltimea=starttimey-starttimea;
        $strobe("Y fall time to A fall time is %f", falltimea/1n);
    end
    cs_a=V(vina);
    cs_y=V(vouty);
    end
Waveform
Alternative Testbench
Universal Stimulus for All Cells

Output waveform from stimulus (from top to bottom) A, B, C, D
Automation Scripting

- Tests a single cell or multiple cells consecutively
- Outputs the result to text file
- Ability to change default parameter values
  - VDD for digital testing
  - Input frequency, mag for analog testing
- Ability to choose simulation time
Script File

- Automatically simulates the test bench
- Steps Include:
  - Establishing which cell is to be tested
  - Creating the Verilog AMS netlist (2 ways)
    - Command “amsdirect”
    - Directly editing the netlist with scripting
  - Elaborating the design: command “ncelab”
  - Simulating the design: command “ncsim”
module TESTBENCH; // Verilog-AMS netlist generated by the AMS netlist, version 5.0.32, USR3.16.35. // Cadence Design Systems, Inc.
include "disciplines.ams"
defparam TESTBENCH.VDD = 5;
module TESTBENCH2 ( f1,OUT1,OUT2,r2,r1,D,C,f2,f3,r3,B,f4,r4,A );
output f1;
input OUT1;
input OUT2;
output r2;
output r1;
output D;
output C;
output f2;
output f3;
output r3;
output B;
output f4;
output r4;
output A;

endmodule
Reported Results

Thu Feb 9 15:20:27 EST 2006

ParryLibTest
ao22x1


rise 1 = 565.200335  fall 1 = 548.881483
rise 2 = 537.356319  fall 2 = 488.049436
rise 3 = 367.078269  fall 3 = 383.547600
rise 4 = 394.282781  fall 4 = 442.307748

ParryLibTest
aoi22x1


rise 1 = 386.208217  fall 1 = 404.994358
rise 2 = 334.311487  fall 2 = 376.908742
rise 3 = 229.695507  fall 3 = 216.506129
rise 4 = 279.239698  fall 4 = 243.309456
Future Work

- Characterization of an Analog Cell
- Harder to do than Digital Cells
  - More parameters to consider
  - Frequency domain analysis
Analog Testbench
DFT in Matlab
Other Analog Characterization

- Frequency Response
  - Check gain for several different frequency inputs until a frequency response is captured

- Non-linearity
  - Put in two frequencies at the input and capture output (1P3)
  - 1dB compression point
Optimization with Scripting

- Iteratively change parameters and simulate until an optimal value is used for:
  - Width/length values
  - Bias current
  - Bias voltage