High Speed Bus Design Using HSPICE Optimization Techniques Based on the Worst Case Design Approach

By
Raj Lakhani, Craig Deutschle, Paul Franzon & Michael Steer
North Carolina State University
Raleigh, North Carolina, 27695

Abstract

In this paper, a deterministic "worst case" design approach based on the HSPICE technique for the interconnect and package optimization of high speed digital circuits is presented. The HSPICE technique is based on using user-defined parameters to control component values and trace dimensions and specifying optimization parameters. To demonstrate the technique, an example of a high speed data bus is presented.

1. Introduction

The "worst case" design approach is based on finding and optimizing the worst possible timing, package, and signal integrity parameters. This approach generally requires complex and efficient algorithms and tends to be computationally intensive.[1]

In this paper, an efficient, multiple parameter "worst case" approach is presented. The approach relies on the accurate modeling of the system transmission lines and package parasitics and the use of the HSPICE optimization routines. When the system is optimized, it is shown that the "worst case" noise and delay is substantially reduced when compared with the circuit optimized using heuristics only.

2. The HSPICE[1] Optimization Technique

The commercial HSPICE package incorporates a powerful linear optimizer for circuit design. With a user-defined optimization program and a known circuit topology, the design components and model parameters are automatically selected to meet DC, AC and transient specifications. The optimization process requires that the user provides the following:

1. The variable parameters and components
2. Minimum and maximum parameter and component limits
3. An initial point to the selected parameter and components
4. The circuit performance goals

Given the optimization specifications, component limits, and initial guess, the optimizer reiterates the circuit simulation until the target electrical specification is met or an optimized solution is found. Since HSPICE employs linear optimizers, the key to a successful optimization strategy is to make a good choice for the initial point.[2]

Bus Design Example:
The example presented in this paper was originally assigned as a class design project. A detailed copy of the circuit schematics and HSPICE code can be found at http://www2.ncsu.edu/eos/project/erl.[3]

The configuration, shown in Figure 1, is an 8-load bus based on drivers and receivers operating on bi-directional pins. The chips used were quad flat packages mounted on a FR-4 PCB with 1/2 oz. copper traces in an SPGS configuration. The topology used was 6 straight bus lines with 4 chips on each side, which meant only one side of the 28 pin quad flat package was used. This side consisted of 6 bi-directional signal pins and a ground pin in the center. The package was modeled with complete LC parasitics, both self and mutual. The mutual capacitances were modeled for up to 2 pins away on both sides. Also, the inputs in each package were delayed by .8ns to reduce Simultaneous Switching Noise (SSN). The packages are connected to the bus using straight stubs which attach to the bus through vias. Both the bus traces and stubs were modeled as lossy transmission lines in the HSPICE code.

[1] Trademark, Meta Software

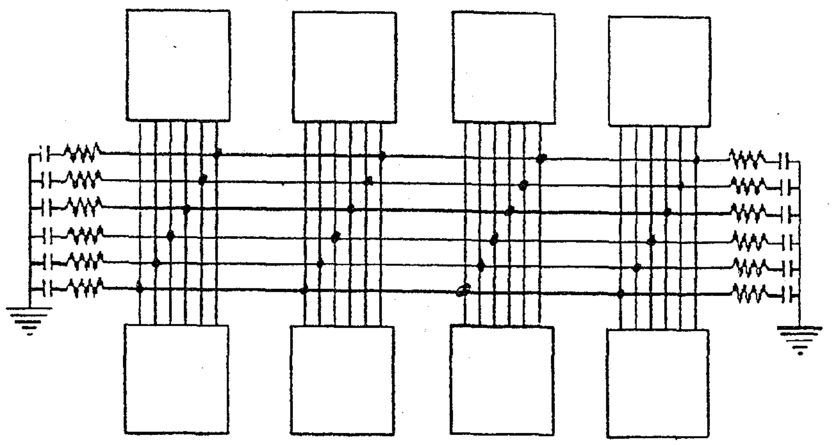
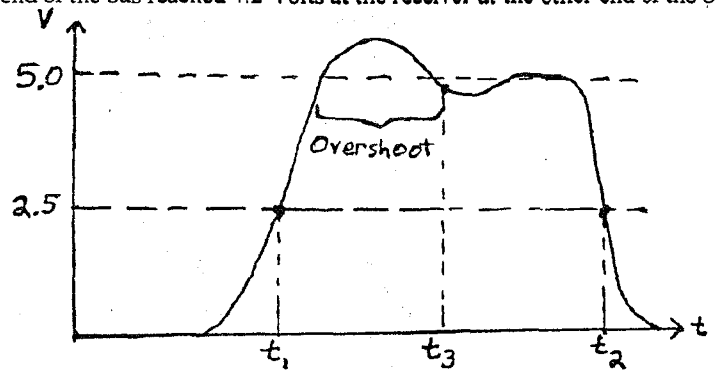


Figure 1.

Extensive, preliminary simulations were carried out in order to determine the initial point for optimization. The initial characteristic impedance of the bus was chosen as 60 ohms. This provided a good trade-off between speed and first-incidence switching. The bus lines were terminated with 60 ohm resistors and 10pF capacitors. This value was determined by extensive simulations with a goal of using a large enough capacitance to overdamp as much noise as possible without affecting delay times on the bus. As a starting point, trace height of .15mm and trace width of .55 were used.[4] To reduce crosstalk, a space-to-height above ground plane ratio of about 2 was assumed. This yielded an initial line spacing of .305mm.
A noise budget was then calculated to determine the design constraints. The drivers and receivers were modeled with "worst case" NMOS and PMOS characteristics. The noise margin for the driver package was found to be about 2.45 Volts. The package was connected to the bus topology to simulate a real load, and an SSN simulation was conducted. After measurements were taken for crosstalk and SSN, the sum of squares approach yielded about 1.4 Volts to give for other noises such as reflection, internal IC, and other unmeasureable noises. [5]
The HSPICE optimization was then conducted using these initial values and constraints.

Optimization:
The goal of this optimization was to minimize noise and delay while varying bus and stub lengths within constrained ranges. The two system parameters that were monitored were overshoot and delay. The critical points that established overshoot are shown in Figure 2. Overshoot was defined as the maximum value between two calculated times. The start of this interval was the time when the rising edge reached 5 Volts. The end of the interval was the halfway point between the rising edge hitting 2.5 Volts and the falling edge hitting 2.5 Volts. Delay was defined as time when a rising edge transmitted from one end of the bus reached 4.2 Volts at the receiver at the other end of the bus.



Overshoot is the maximum value from t_1 to t_3.

$t_3 = \frac{t_1 + t_2}{2}$

Figure 2.

The proper order for HSPICE optimization statements is:[2]
1. Analysis statement with OPTIMIZE
2. .MEASURE statement specifying optimization goals
3. Ordinary analysis statement
4. Output statements

A transient analysis statement was specified with the OPTIMIZE statement and the name of the optimization. The RESULTS command allowed for the system goals to be specified, and the name of the optimization model was given in the MODEL statement. The OPTIMIZE statement was:

```
.tran 1ns 100ns SWEEP OPTIMIZE=opt1 RESULTS=rise_t,
    + undershoot1, overshoot2 MODEL=optmodel
```

The optimization name is included in the model section to specify it as an optimization:

```
.MODEL optmodel OPT
```

The .MEASURE statements were included to try to measure and minimize overshoot as follows:

```
.measure tran osh1 when v(p8_1)='2.5' cross=1
.measure tran osh2 when v(p8_1)='2.5' cross=2
```

These statements store the times when the voltage at node p8_1 is 2.5V. The "cross" parameter simply means when the voltage crosses 2.5V, so osh1 would be on the rise, and osh2 would be on the fall.

```
.measure tran tmid_o param='(osh1+osh2)/2'
.measure tran vmid_o find v(p8_1) AT='tmid_o'
.measure tran o_from when v(p8_1)='vmid_o' rise=1
```

The first of these statements finds the midpoint between osh1 and osh2 and stores it in tmid_o. Next, the voltage at the middle of the wave is found at t=tmid_o. This value is presumed to be 5.0V. Finally, the time instance on the first rise that is equal to vmid_o is stored in o_from. Now that a finite time interval is set, goals can be established. In this case, the maximum value was set equal to 5 Volts, resulting in an unachievable overshoot goal of 0 Volts:

```
.measure tran overshoot1 MAX v(p8_1) from='o_from' to='tmid_o' GOAL=5
.measure tran undershoot1 MIN v(p8_1) from='o_from' to='tmid_o' GOAL=5
```

Another example of this configuration is to measure negative overshoot after the falling edge is given in the HSPICE code. This was similarly accomplished by finding a minimum value in a time interval beginning when the falling edge first hits 0 Volts. All of the measurements are written to a .MT0 file. Undershoot goals could be specified similarly. However, in this design undershoot levels were minimal.

After the .MEASURE section, the parameters to be optimized and the ranges to be explored are entered:

.PARAM *parameter*=OPTxxx (initial_value, minimum_value, maximum_value)

```
.PARAM height=opt1(.1524m, .1m, .2m)
.PARAM width=opt1(.54864m, .2m, 5m)
.PARAM space=opt1(.3048m, .15m, 5m)
.PARAM ro=opt1(57, 55, 60)
.PARAM co=opt1(10p, 10f, 30p)
```

Once something is described in the parameter section of the circuit file, it can be used again and again without editing. An example of this would be as follows:

```
.PARAM height=.15m
.PARAM width=200u
.PARAM space=305u
```

What this essentially does is define the initial values of the bus height, width, and spacing so they can be referenced again like constants.

The goal of minimizing delay was specified by asking the optimizer to aim at the (unachievable) goal of making all the signals transition together. If both signals hit a voltage level at the same time, the delay was zero. The rise time at the input of the far end receiver was set to a goal of .7ns. This is equal to the time when the original transmitted wave hit 4.2 Volts, thereby minimizing delay. To incorporate rise time as an optimization factor, the following line was used:

```
.measure tran rise_t when v(p8_1)='4.2' cross=1 GOAL=.7ns
```

If certain parameters are more important than others, a WEIGHT=x statement can be added at the end of the .MEASURE line. HSPICE will optimize the highest weight number with closer accuracy.

Finally, analysis section was set up to run the normal transient analysis:

```
.temp 25
.tran 1ns 100ns
```

3. Results:

After running HSPICE with the initial values, the following optimized values were obtained:

width=.2 mm
height=.15 mm
space=.305 mm
R_0=60 Ω
C_0=10 pF
Z_0=60 Ω

These changes resulted in an 80mV reduction in noise on a quiet line, 100mV reduction of noise on an active line, and significant reduction in delay times when compared with the heuristically optimized initial design. This is shown in Figure 3.

The HSPICE optimization technique allowed for an accurate, time-domain, "worst case" analysis of a high speed bus. By establishing parameters to be varied, and assigning goals that were monitored, an optimum design was achieved. By varying the speed of the inputs and length of the bus, multiple optimizations runs resulted in a wire rule generation graph that plotted maximum stub lengths versus bus speed and length. The average CPU time required for each optimization was 20 hours on a Sun Sparc 20. This time was greatly dependent on the accuracy of the initial guess for each parameter.

4. Conclusions

We have shown how to use the HSPICE linear optimizer to solve a classical signal integrity problem of Bus Optimization. The keys to success include (1) finding an initial design point that is likely to lead to an optimization in gradient descent and (2) creative use of the HSPICE '.MEASURE' statements to express the signal integrity goals.

5. References

[1] Tawfik Arabi & Mike Tripp, "A Numerical Frequency Domain Technique For The Optimization of High Speed Multi-Conductor Transmission Line Circuits Based on the Worst Case Design Approach." **Intel Paper.**, Vol. X 1995, pp. 7-9.

[2] Meta Software, "**HSPICE User's Manual.**" Vol. 2 1992.

[3] http://www2.ncsu.edu/eos/project/erl

[4] Howard V. Johnson & Martin Graham, "**High Speed Digital Design.**" Prentice Hall. Englewood Cliffs, New Jersey, 1993.

[5] Paul D. Franzon. "**ECE 544 Lecture Notes.**" 1996.

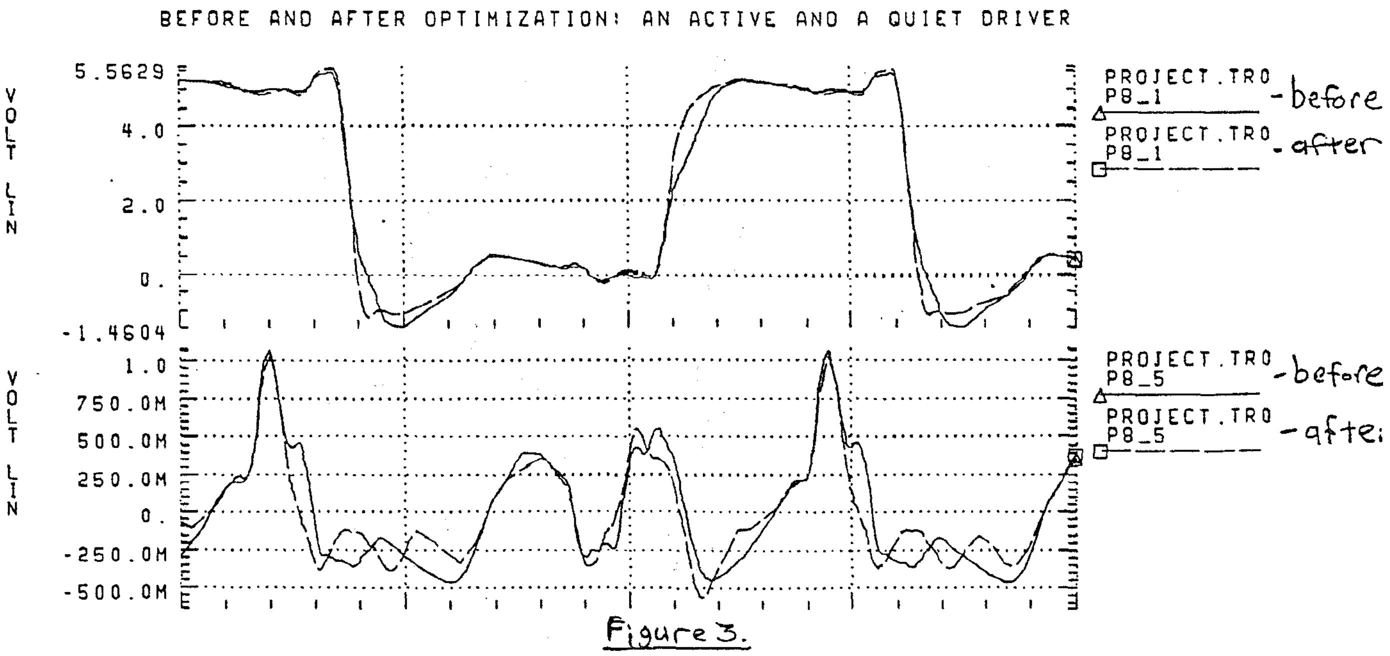


Figure 3.