

THE DESIGN VERIFICATION MODULE FOR A SAW DESIGN AUTOMATION SYSTEM

M. J. McCOLLISTER and S. M. RICHIE

University of Central Florida, Department of Electrical and Computer Engineering, Orlando, Florida 32816-2450

ABSTRACT

This paper presents a stand-alone version of the design verification module for a SAW design automation system. The automation system that is being developed requires a verification phase that will be used to examine S-parameter data from test devices and determine if a specific device meets design requirements. This paper includes the specification and performance data structures, methods of extracting performance data, and methods of verifying performance versus requirements. The algorithms and techniques for efficiently extracting device performance data from experimental data are presented. Techniques including Fourier and wavelet coefficient reduction for parameter extraction are included. The core of the system contains the specification and performance data structures appropriate for SAW bandpass filter devices. The specifications contain the time response constraints, frequency response magnitude and phase, and delay constraints. Performance data is extracted from measured or calculated S-parameters as a function of frequency. This data is compared with the design requirements to see if a device meets these specifications. If a device fails, the device parameters that caused the failure are identified. All performance failures can be used as a basis for design corrections. The algorithms and programs developed in this paper will be available via the Internet at <http://pegasus.cc.ucf.edu/~mjm05082/fcs1999/>.

1. INTRODUCTION

The development of a SAW design automation system includes four main parts. These parts are the user specification, rule and model based SAW design, analysis and layout, and performance evaluation and verification [1]. While the end result of a SAW design automation system is the design of a working SAW device, the system must verify that the data derived meets the designer's specifications.

This paper presents the development efforts of SawVerify, the stand-alone version of a SAW design verification module. While this program does not design a SAW device given user specifications, it does take measured data from a real SAW device and extracts various parameters and determines which parameters meet the user specifications. A future completed automated SAW design system will use modeled data as well as measured data from a physical device to determine if a SAW device meets specifications.

2. DATA STRUCTURES

SawVerify was developed using Borland C++ Builder 4 running on Microsoft Windows 95/98. The

object oriented paradigm of C++ provided to be the natural way of representing data associated with a measured device. A class, called TMeasDev, was created to encapsulate the data and derived parameters from raw S-parameter data. This class fully represents the SAW device as a "black box". By doing so, the class can either encapsulate data from a measured device or from a device model. This class also separates the GUI from the data and parameter extraction functions so that porting to another compiler requires very little changes to the core processing code.

2.1 C++ and the Standard Template Library

C++ includes the standard template library (STL) which contains predefined classes for complex numbers, vectors and many other data structures. The STL vector class was used extensively, instead of the ANSI C arrays, to simplify memory allocation and vector sizing. This also reduced errors and development time.

2.2 The TMeasDev Class

The TMeasDev class was designed to not only extract parameters from a device, but also to verify that the device meets user specifications. The data sizes necessary to contain the device data of N points include three float vectors of size N, four complex vectors of size N, and one two-dimensional complex vector of size 4xN. Temporary vectors include a complex twiddle array of size N and two float vectors of size N. A small number of local variables are also needed to store values for individual parameters.

Because of the verification, a class called TVals was created to encapsulate the minimum, typical, maximum, and measured parameter values. The TVals class is nested within the TMeasDev and is isolated from the

```
class TMeasDev { // highly truncated class
private: // User declarations
    String ImportedFileName;
    vector<Float> RawFreq, Time, WorkFreq;
    vector<Complex> Delay, WorkSParams,
                    Impulse, SmothedMag;
    vector< vector<Complex> > RawSParams;
public: // User declarations
    TMeasDev(); // constructor
    void SetImportedFileName(String& fn);
    String GetImportedFileName();
    const vector<Float>& GetRawFreq();
    const vector<Float>& GetTime();
    const vector<Float>& GetWorkFreq();
    const vector<Complex>& GetDelay();
    const vector<Complex>& GetWorkSParams();
    const vector<Complex>& GetImpulse();
    const vector< vector<Complex> >&
                    GetRawSParams();
    bool ImportRawData(ifstream& infile);
};
```

Figure 1: TMeasDev Class Fragment

programmer. In order to model such a device, arrays are used to store raw data as well as processed data. None of these arrays are directly accessible to the user and require the user of class member function to access or modify them. *Figure 1* shows a fragment of the TMeasDev class, which holds the data. The ImportRawData() function is used to fill the internal arrays and the various “Get” functions are used to access the data for analysis or plotting.

3. PERFORMANCE DATA EXTRACTION

The current system uses measured S-parameter data and calculates derived performance parameters. S-Parameter data is read from an ASCII text file in any format be it real and imaginary, linear or logarithmic, degrees or radians. None of the user specification data is used in determining the measured parameters, meaning that SawVerify will extract parameters strictly from raw data. Measured parameters of interest include bandwidth, center frequency, lower and upper bandedges, insertion loss at center frequency, minimum insertion loss, magnitude slope, magnitude ripple, center frequency delay, phase ripple and phase maximum deviation from linear. Also the time domain response is calculated and presented.

If SawVerify is presented with raw S-parameter data that has significant RF feedthrough and needs to be windowed, then the user has the ability to time gate the data to remove the RF feedthrough and/or window it using select window types. *Figure 2* shows the S21 raw magnitude response of Carter’s 13 chip uniformly weighted stepped chirp code SAW device [2].

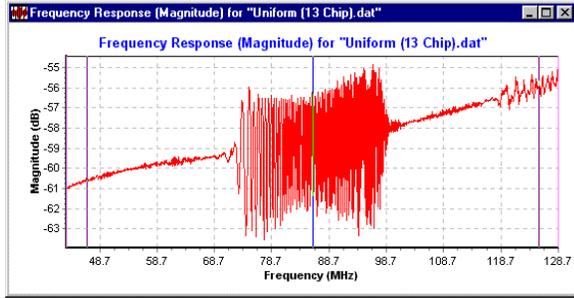


Figure 2: Raw Magnitude Data

By applying the Hanning window and time gating the raw data, the response shown in *Figure 3* is obtained.

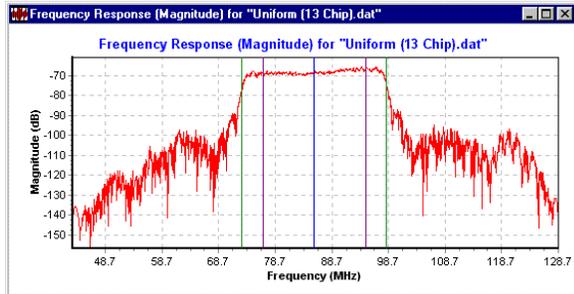


Figure 3: Time Gated and Hanning Window

All of the data in this paper refers to Carter’s 13 chip uniformly weighted stepped chirp code SAW device. This device displays the properties of a high loss bandpass SAW filter with high RF feedthrough and was perfect to test the techniques presented in this paper.

3.1 First Guess for Upper and Lower Bandedges

Initial investigation found the process of calculating the upper and lower bandedges to be more difficult than expected. While it is easy for a human to visually detect the bandpass region, it is not trivial to code an algorithm to do the same, especially with high loss data and high level of RF feedthrough. However, a simple technique, called the top hat method, performs a least-squared error calculation of a pulse function (or top hat) with the magnitude data. This algorithm correlates all possible pulse widths and positions with the data. *Figure 4* shows simplified magnitude data, X , with one possible top hat function, g .

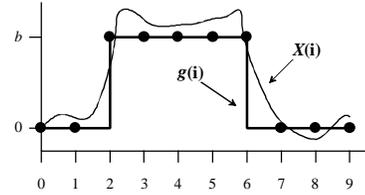


Figure 4: First Guess Bandpass Region Using Top Hat

This technique simply takes the least-squared error of the vector representing the magnitude data, X , and the vector representing the top hat, g . *Equation (1)* shows the dot product of the difference between the two vectors that has to be minimized:

$$f = (X - bg)^T \cdot (X - bg). \quad (1)$$

By taking the partial derivative of f ,

$$\frac{\partial f}{\partial b} = -2g^T X + 2bg^T g, \quad (2)$$

and setting it equal to zero, b is found to be

$$b = \frac{g^T X}{g^T g}. \quad (3)$$

Factoring *Equation (1)* and substituting *Equation (3)* for b , yields:

$$f = XX^T - \frac{(g^T X)^2}{g^T g}. \quad (4)$$

To minimize f , then

$$\frac{(g^T X)^2}{g^T g} \quad (5)$$

is maximized to reduce the error and thus finds the best first guess. In other words pick the top hat function, g , that returns the maximum value for *Equation (5)*.

An efficient implementation of *Equation (5)* uses a partial sum vector:

$$S_k = \sum_{i=0}^{N-1} X_i. \quad (6)$$

Since $g^T X$ is the difference of partial sums and $g^T g$ is the top hat width, then Equation (5) can be rewritten as:

$$\frac{(S_j - S_i)^2}{j - i}, \quad (7)$$

which is computationally efficient. The values for i and j from the maximum value of Equation (7) give a good first guess position of the lower and upper bandedges.

Once the top hat method determines the first guess upper and lower bandedges, a simple algorithm is used to find the 3 dB bandwidth, or the bandwidth at any level as specified by the user. The outer vertical lines on Figure 3 show the 10 dB.

3.2 Other Parameters

Most other device parameters are easily extracted from this point on. For example, the center frequency delay is obtained by calculating the slope of the phase at center frequency. The center frequency delay calculated from the phase response of Carter's chirp SAW filter is shown as the vertical line in the middle of the time domain impulse response in Figure 5.

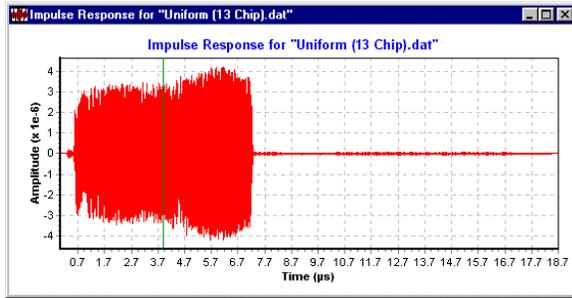


Figure 5: Time Domain Impulse Response

The most difficult parameter to be extracted was the magnitude slope. Many investigated algorithms have a very difficult time finding the local maximums of a "noisy" magnitude response. Wavelet and FFT smoothing techniques were investigated to determine if coefficient reduction of the magnitude response aided in extracting the local maximums of the bandpass region.

4. ALGORITHM COMPARISONS

A wavelet transform is similar to the Fourier transform in that it uses a series of orthogonal functions to transform a function to an alternative domain, as shown by the basic wavelet recursion equation [3] in Equation (8):

$$\psi(t) = \sum_n h(n) \sqrt{2} \psi(2t - n). \quad (8)$$

Where the scaling coefficients are calculated as follows:

$$h(n) = \sqrt{2} \int \psi(t) \psi(2t - n) dt. \quad (9)$$

Equation (8) and Equation (9) show how similar the wavelet transform is to the Fourier transform. The difference is that the wavelet transform uses finite basis functions as opposed to the infinitely oscillating sinusoidal functions used in the Fourier transform [4]. The wavelet transform breaks a function into sections of scale. By removing all but the coefficients that correspond to the high scale (low detail) representation of a function then data can be smoothed yet still retain the overall shape of the data. This is a simplistic method of denoising as presented by Burrus [3].

4.1 Wavelet Smoothing Techniques

There are two methods of coefficient reduction investigated, truncation and thresholding. Truncation is simply removing all but the first N wavelet coefficients. This retains the low detail information. The threshold method removes all but the N largest coefficients thus possibly retaining some higher detail information. Figure 6 shows a portion of the Daubenchies-20 wavelet transform of Carter's SAW device. The vertical line represents the truncation point in which the first thirty-two wavelet coefficients are used. The horizontal line shows the threshold in which the thirty-two largest coefficients are kept. This uses some higher detail coefficients that the simple truncation did not use.

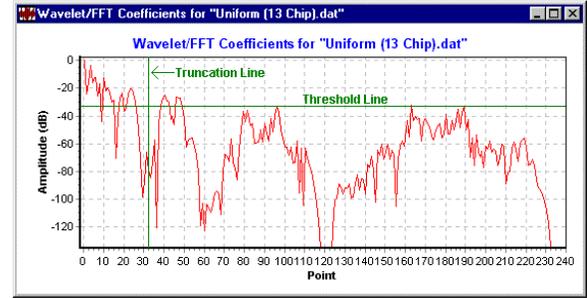


Figure 6: Carter's Daubenchies-20 Wavelet Coefficients

4.2 Wavelet Smoothing Results

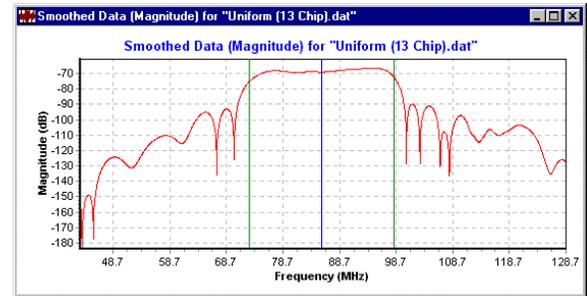


Figure 7: Smoothed, First 32 Daub-20 Wavelet Coef.

Figure 7 shows the smoothed magnitude response of Carter's chirp code SAW device. Smoothing was performed by transforming the magnitude response to the wavelet domain using the Daubenchies-20 wavelet transform [5]. Then all but the first thirty-two coefficients were zeroed. This truncation removed the

low scale (or high detail) wavelet coefficients. Finally the now truncated wavelet coefficients were transformed back to the magnitude domain.

Compare *Figure 3* with *Figure 7* to see that the general shape of the magnitude response is retained. This smoothed data is much easier to extract the local maximums so that magnitude slope can be calculated from the raw magnitude data.

The Daubenchies-20 wavelet is a very smooth wavelet, much like a damped $\sin(x)/x$ function. Truncating all but thirty-two coefficients using a lower order and “rough” wavelet basis function, such as the Daubenchies-4, results in a not so smoothed magnitude response as shown in *Figure 8*. This smoothed version of the magnitude response creates artificial inflection points because the basis function is very rough. This is not a desired result.

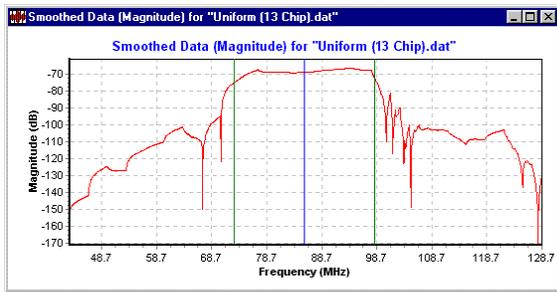


Figure 8: Smoothed, First 32 Daub-4 Wavelet Coef.

The same truncation method was applied using a standard FFT algorithm [6]. The results were very similar to the Daubenchies-20 but faired a little better as shown in *Figure 9*. After testing the wavelet and FFT techniques on various SAW devices from Carter as well as some commercially available devices, it was found that the FFT smoothing method using truncation was the best. Thresholding kept some high frequency or high detailed components that interfered with the algorithm that detected the local maximums of the edges of the bandpass region.

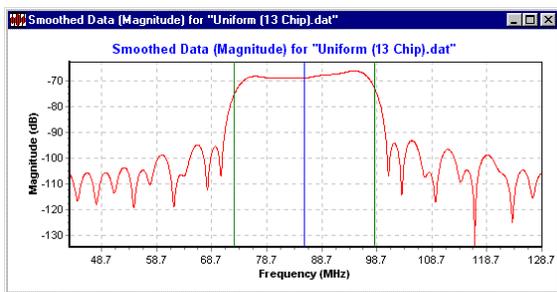


Figure 9: Smoothed, First 32 FFT Coefficients

Parameters from commercially produced SAW filters have been extracted and due to the low loss and low RF feedthrough, the parameter extraction did not require the FFT or wavelet coefficient reduction, as did the high loss filters. SawVerify by default does not use FFT or wavelet coefficient reduction if the minimum insertion loss is less than 40 dB.

5. VERIFICATION METHODS

Verification of a SAW device proved to be quite simple. Because the TVals class contained minimum, typical and maximum user constrains, it was very simple to compare these to the extracted values. For minimum and maximum values, the device parameters that caused the failure are identified “red”. However, a percentage range is used to signify how good the typical value is. If it is within 1% then the value is good; if it is within 5% then the value is marked “yellow”; outside 5% the value is marked “red”. *Figure 10* illustrates how this is presented to the user.

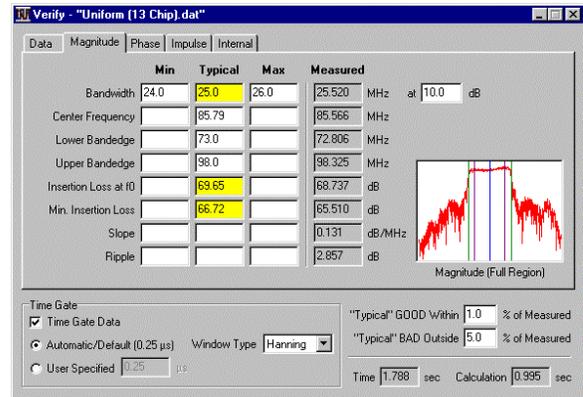


Figure 10: Magnitude Response Information

6. CONCLUSION

A stand-alone version of the design verification module for a SAW design automation system was presented. Methods for extracting device parameters were presented. The top hat method determined a good starting point for parameter extraction. FFT coefficient reduction using truncation proved to be very helpful with magnitude slope extraction. A simple method of presenting parameter failure was also presented.

7. REFERENCES

- [1] M. J. McCollister and S. M. Richie, “The Development of a First Pass Verification Module for a SAW Filter Design Automation System”, *IEEE Ultrasonics Symposium Proceedings*, vol. 1, pp. 133-136, 1991.
- [2] S. E. Carter, “Development and SAW Device Implementation of a New Weighted Stepped Chirp Code Signal for Direct Sequence Spread Spectrum Communications Systems”, Ph. D. Dissertation, University of Central Florida, Orlando, Florida, 1998.
- [3] C. S. Burrus, R. A. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*, New Jersey: Prentice-Hall, ch. 10, pp. 205-211.
- [4] M. V. Wickerhauser, *Adapted Wavelet Analysis from Theory to Software*, New York: IEEE Press, 1994.
- [5] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Second Edition, Cambridge: Press Syndicate, 1992, ch. 13.10, pp. 591ff.
- [6] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing*, NY: Macmillan, 1988, ch. 9, pp. 698-721.