International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:3, No:4, 2009

Symbolic Analysis of Large Circuits Using Discrete Wavelet Transform

Ali Al-Ataby , Fawzi Al-Naima

***Abstract*—**Symbolic Circuit Analysis (SCA) is a technique used to generate the symbolic expression of a network. It has become a well-established technique in circuit analysis and design. The symbolic expression of networks offers excellent way to perform frequency response analysis, sensitivity computation, stability measurements, performance optimization, and fault diagnosis. Many approaches have been proposed in the area of SCA offering different features and capabilities. Numerical Interpolation methods are very common in this context, especially by using the Fast Fourier Transform (FFT). The aim of this paper is to present a method for SCA that depends on the use of Wavelet Transform (WT) as a mathematical tool to generate the symbolic expression for large circuits with minimizing the analysis time by reducing the number of computations.

***Keywords*—**Numerical Interpolation, Sparse Matrices, Symbolic Analysis, Wavelet Transform.

I. Introduction

SCA refers to the calculation of network functions where all or part of the circuit parameters are represented by symbols. The applicability of symbolic simulation techniques to the analysis and synthesis of analog integrated circuits has been known for a long time [1,2,3]. SCA has been developed to help designers get a better understanding of circuit behaviours using the symbolic expressions for the circuit performances. This technique is quite mature in analysis of linear circuits [4,5].

Given the exponential increase in complexity and the time required to do SCA with the circuit size, finding a method that can handle large circuits keeping both the complexity and time as minimum as possible is a challenging factor [5].

SCA methods (in a close connection with numerical methods) can be divided mainly into two categories. These are the *topological* and the *numerical* methods [6]. Each one of these methods has its own advantages and limitations. For instance, in topological methods the number of elements represented as symbols is large but the circuits that can be handled is small. On the other hand, in numerical methods, fairly large networks can be handled but the number of symbolic variables is limited. The numerical interpolation method constitutes a very efficient technique for the calculation of network function coefficients with only the complex frequency in symbolic form [6]. The direct application of numerical interpolation method can be used to solve problems of system matrix size of around 30 and about 10 elements only represented as variables beside the complex frequency "s" [7,8]. Modified versions of numerical interpolation method are available and proved to be efficient; however, it still suffers from serious limitation in practice, which is the rapidly increasing amount of calculation as the number of symbols to be handled increases. This naturally leads to escalation of computer CPU time and memory requirements, and hence, the famous overflow problem.

Looking to the issue from linear system window, it is so obvious that the size of the matrices under processing needs to be reduced. This can be done naturally by two methods: approximation (omission of insignificant terms in the system matrix) [5] and compression (introducing sparsity to increase the number of zeros in the system matrix). Following this strategy, larger circuits can be analysed with less computation efforts.

This paper follows the second method, i.e. compressing the system matrix by introducing sparsity. To achieve this aim, a clever and promising mathematical transform will be used. This transform is the Discrete Wavelet Transform (DWT). To simulate the application to SCA, a program was written and tested using MATLAB.

The next sections will describe the numerical interpolation traditional method using FFT, then the DWT will be introduced and its use as a method to compress system matrices will be illustrated. Some experimental results will be shown, and the paper will be concluded with comments on the proposed method.

II. Numerical Interpolation Method For Symbolic Analysis

Numerical interpolation methods are based on the theory and implementation of numerical methods for generating symbolic functions of networks. They seem to have a lower computational cost than other well-known symbolic analysis algorithms such as *parameter extraction method***.**

The following discussion will introduce the idea of using interpolation in finding network transfer functions using the Discrete Fourier Transform (DFT) in interpolation [8, 9,10, 11].

A. Polynomial Interpolation

First, N+1 points will be found by evaluating the function:

$$P_N(x) = \det[A(x)] \quad (1)$$

at x_0, x_1, ..., x_N, where N is the maximum power of x. Now, there are N+1 distinct points (x_i, $y_i=P_N(x_i)$), i=0, 1, ..., N. Both

Ali Al-Ataby was with the Department of Computer Engineering, College of Engineering, Nahrain University, PO Box 64040, Baghdad, Iraq. He is now with the Department of Electronics and Electrical Engineering, Liverpool University, Liverpool, UK (email: AliAlAtaby@liverpool.eng.uk.com).

Fawzi Al-Naima is with the Department of Computer Engineering, College of Engineering, Nahrain University, PO Box 64040, Baghdad, Iraq. (Corresponding author, email: fawzi_cmc@yahoo.com).

x_i and y_i may be real or complex numbers. The coefficients of the polynomial:

$$P_N(x) = \sum_{n=0}^{N} a_n x^n \tag{2}$$

are required to be found such that it passes through the given set of points.

Inserting x_i into the polynomial (2), the following set of equations will be obtained:

$$a_0 + a_1 x_i + a_2 x_i^2 + ... + a_N x_i^N = y_i \ , i = 0, 1, ..., N \tag{3}$$

with unknowns a_0, a_1, a_2, ..., a_N. Since there are N+1 unknown coefficients and the same number of equations, a matrix equation can be formulated as follows:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & ... & x_0^N \\ 1 & x_1 & x_1^2 & ... & x_1^N \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & ... & x_N^N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{bmatrix} \tag{4}$$

Or:

$$[X][A] = [Y] \tag{5}$$

The solution of (5) provides the unknown coefficients.

Since there is a choice of selecting the points x_i, the question arises as to what the choice should be in order to obtain the best possible result. It can be shown that the interpolation with real x_i is, in general, numerically unstable [10].

B. The use of Discrete Fourier Transform (DFT)

The DFT interpolation can be derived by introducing a special symbol for the matrix X in (5):

$$X = \left[x_i^n\right] \tag{6}$$

where the index i and the exponent n run from 0 to N. If the set of points x_i is chosen to be uniformly spaced on the unit circle in the complex plane, then these points are:

$$x_0 = 1, \ x_k = \exp\left[\frac{j2k\pi}{N+1}\right], k = 1, 2, ..., N \tag{7}$$

Introducing the substitution:

$$w = \exp\left[\frac{j2\pi}{N+1}\right] \tag{8}$$

then:

$$x_k = w^k \tag{9}$$

and:

$$X = \left[w^{in}\right] \tag{10}$$

It can be shown that [5]:

$$X^{-1} = \frac{1}{N+1}\left[w^{-in}\right] = \frac{1}{N+1} X^* \tag{11}$$

Where X^* denotes the transpose conjugate matrix and i runs from 0 to N.

The solution of (5) with the points defined by (7) is:

$$A = X^{-1}Y = \frac{1}{N+1}\left[w^{-in}\right] Y \tag{12}$$

or:

$$a_n = \frac{1}{N+1}\sum_{k=0}^{N} y_k \ w^{-nk} \ , n = 0, 1, ..., N \tag{13}$$

The original polynomial in (2), evaluated at x_k, can be written as:

$$y_k = \sum_{n=0}^{N} a_n \ w^{nk} \tag{14}$$

Equations (13) and (14) represent the solution of one another. They are called the DFT pair.

To improve the speed of the method, a fast algorithm in interpolation can be used. Algorithms that reduce the computational cost of DFT are called in general the Fast Fourier Transform (FFT). The DFT has been studied extensively and it can be programmed in a very efficient way, particularly when $N+1=2^m$, m being a positive integer. The number of operations required in this case is m×(N+1) [8,12].

III. The Use Of The Wavelet Transform

In this section, the use of the Discrete Wavelet Transform (DWT) in the area of SCA will be illustrated. Before that, the DWT will be explained briefly.

A. The Discrete Wavelet Transform (DWT)

Like the FFT, the Discrete Wavelet Transform (DWT) is a fast linear operation that operates on a data vector whose length is an integer power of two, transforming it into a numerically different vector of the same length. Also, like the FFT, the DWT is invertible and in fact orthogonal, that is, the inverse transform when viewed as a big matrix is simply the transpose of the transform. Both FFT and DWT, therefore, can be viewed as a rotation in space, from the input space (or time) domain, where the basis functions are the unit vectors e_I, or Dirac delta functions in the continuum limit, to a different domain. For the FFT, this new domain has basis functions that are the familiar sines and cosines. In the wavelet domain, the basis functions are somewhat more complicated and have the fanciful names "*mother functions*" and "*wavelets*" [12,13,14,15].

Of course there are infinity of possible bases for function space, almost all of them uninteresting. What makes the wavelet basis interesting is that, unlike sines and cosines, individual wavelet functions are quite localized in space; simultaneously, like sines and cosines, individual wavelet functions are quite localized in frequency or (more precisely) characteristic scale. The particular kind of dual localization achieved by wavelets renders large classes of functions and operators sparse, or sparse to some high accuracy, when transformed into the wavelet domain. Analogously with the Fourier domain, where a class of computations, like convolutions, become computationally fast, there is a large class of computations (those that can take the advantage of

sparsity) that become computationally fast in the wavelet domain [13,14,9].

Unlike sines and cosines, which define a unique Fourier transform, there is not one single unique set of wavelets; in fact, there are infinitely many possible sets. Roughly, the different sets of wavelets make different trade-offs between how compactly they are localized in space and how smooth they are.

The wavelet transform procedure is to adopt a wavelet prototype function, which is the mother function. Temporal analysis is performed with a contracted, high-frequency version of the prototype, while frequency analysis is performed with a dilated, low-frequency version of the same wavelet. Because the original signal or function can be represented in terms of a wavelet expansion (using coefficients in a linear combination of the wavelet functions), data operations can be performed using the corresponding wavelet coefficients. And if the best wavelets adapted to the data are further chosen, or truncate the coefficients below a threshold, the data is sparsely represented. This sparse coding makes wavelet an excellent tool in the field of data compression. It is this feature that will make the WT plays as a good tool to speed up the SCA problem.

B. Daubechies Wavelet Filter Coefficients

A particular set of wavelets is specified by a particular set of numbers, called *wavelet filter coefficients*. Here, the wavelet filters that will be followed are the ones discovered by Daubechies [15]. This class includes members ranging from highly localized to highly smooth. The simplest (and most localized) member, often called *DAUB4*, has only four coefficients, c_0, c_1, c_2, *and* c_3 [14,15].

Consider the following transformation matrix acting on a column vector of data to its right:

$$W = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & & & & & & & & \\ c_3 & -c_2 & c_1 & -c_0 & & & & & & & & \\ & & c_0 & c_1 & c_2 & c_3 & & & & & & \\ & & c_3 & -c_2 & c_1 & -c_0 & & & & & & \\ \vdots & \vdots & & & & & & & & & & \\ \vdots & \vdots & & & & & & \ddots & & & & \\ \vdots & \vdots & & & & & & & & & & \\ & & & & & & c_0 & c_1 & c_2 & c_3 & & \\ & & & & & & c_3 & -c_2 & c_1 & -c_0 & & \\ c_2 & c_3 & & & & & & & c_0 & c_1 & & \\ c_1 & -c_0 & & & & & & & c_3 & -c_2 & & \end{bmatrix} \tag{15}$$

Here, blank entries signify zeroes. Note the structure of this matrix. The first row generates one component of the data convolved with the filter coefficients c_0, c_1, c_2, and c_3. Likewise the third, fifth, and other odd rows. If the even rows followed this pattern, offset by one, then the matrix would be a circulant, that is, an ordinary convolution that could be done by FFT methods (Note how the last two rows wrap around like convolutions with periodic boundary conditions.). Instead of convolving with c_0, c_1, c_2, and c_3, however, the even rows perform a different convolution, with coefficients c_3, $-c_2$, c_1, and $-c_0$. The action of the matrix, overall, is thus to perform two related convolutions, then to decimate each of them by half (throw away half the values), and interleave the remaining halves [14].

It is useful to think of the filter c_0, c_1, c_2, and c_3 as being a smoothing filter, called H, something like a moving average of four points. Then, because of the minus signs, the filter c_3, $-c_2$, c_1, and $-c_0$, called G, is not a smoothing filter (In signal processing contexts, H and G are called *quadrature mirror filters* [14]). In fact, the c's are chosen so as to make G yield, insofar as possible, a zero response to a sufficiently smooth data vector. This is done eventually by requiring the sequence c_3, $-c_2$, c_1, and $-c_0$ to have a certain number of vanishing moments. When this is the case for p moments (starting with the zeroth), a set of wavelets is said to satisfy an "approximation condition of order p". This results in the output of H, decimated by half, accurately representing the data's "smooth" information. The output of G, also decimated, is referred to as the data's "detail" information.

For such a characterization to be useful, it must be possible to reconstruct the original data vector of length N from its N/2 smooth or s-components and its N/2 detail or d-components. That is effected by requiring the matrix (15) to be orthogonal, so that its inverse is just the transposed matrix:

$$W^{-1} = W^T = \begin{bmatrix} c_0 & c_3 & & & & \cdots & \cdots & & & c_2 & c_1 \\ c_1 & -c_2 & & & & & & & & & \\ c_2 & c_1 & c_0 & c_3 & & & & & & & \\ c_3 & -c_0 & c_1 & -c_2 & & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & c_2 & c_1 & c_0 & c_3 & & & \\ & & & & c_3 & -c_0 & c_1 & -c_2 & & & \\ & & & & & & c_2 & c_1 & c_0 & c_3 & \\ & & & & & & c_3 & -c_0 & c_1 & -c_2 & \end{bmatrix} \tag{16}$$

Now, since:

$$W\,W^{-1} = W\,W^T = I \tag{17}$$

Where I is the identity matrix, one sees immediately that matrix (16) is the inverse of matrix (15) if and only if the following two equations hold:

$$\begin{aligned} c_0^2 + c_1^2 + c_2^2 + c_3^2 &= 1 \\ c_2 c_0 + c_3 c_1 &= 0 \end{aligned} \tag{18}$$

If additionally, the approximation condition of $p = 2$ is required, then the following two additional relations must be true:

$$\begin{aligned} c_3 - c_2 + c_1 - c_0 &= 0 \\ 0c_3 - 1c_2 + 2c_1 - 3c_0 &= 0 \end{aligned} \tag{19}$$

Equations (18) and (19) are 4 equations for 4 unknowns c_0, c_1, c_2, and c_3, first recognized and solved by Daubechies. The unique solution (up to a left-right reversal) is:

$$c_0 = (1+\sqrt{3})/4\sqrt{2} \qquad c_1 = (3+\sqrt{3})/4\sqrt{2}$$
$$c_2 = (3-\sqrt{3})/4\sqrt{2} \qquad c_3 = (1-\sqrt{3})/4\sqrt{2} \tag{20}$$

In fact, DAUB4 is only the most compact of a sequence of wavelet sets: If we had six coefficients instead of four, there would be three orthogonality requirements in equation (18) (with offsets of zero, two, and four), and requiring the vanishing of p = 3 moments in equation (19). In this case DAUB6 is obtained and the solution coefficients can also be found by following the same steps.

For higher p, up to 10, Daubechies has tabulated the coefficients numerically [14,15]. The number of coefficients increases by two each time p is increased by one.

C. The Use of DWT for Fast Solution of Linear Equations

One of the most interesting and promising wavelet applications is linear algebra [14]. The basic idea is to think of an integral operator (that is, a large matrix) as a digital image. Suppose that the operator compresses well under a two-dimensional wavelet transform, i.e., that a large fraction of its wavelet coefficients are so small as to be negligible. Then any linear system involving the operator becomes a *sparse system* in the wavelet basis. In other words, to solve:

$$A \,.\, x = b \tag{21}$$

Then, wavelet-transform the operator A and the right-hand side b by:

$$\tilde{A} \equiv W \,.\, A \,.\, W^T \quad , \quad \tilde{b} \equiv W \,.\, b \tag{22}$$

Where W represents the one-dimensional wavelet transform and W^T is the transpose (or inverse) of W, then solve:

$$\tilde{A} \,.\, \tilde{x} = \tilde{b} \tag{23}$$

which is a *sparse system* in the wavelet basis. This property can be used to solve the linear system in a faster way (due to less computation overhead) than the normal numerical techniques including the FFT. By solving the obtained sparse system, the solution can obtain almost in a real time basis [14].

Finally, transform to the answer by the inverse wavelet transform:

$$x = W^T \,.\, \tilde{x} \tag{24}$$

The result will appear with a high accuracy as compared with the use of other transforms to perform the same task [14].

The method discussed above was implemented and verified for solving numerical linear systems in a fast and compressed way. It is also adopted to solve the linear system that will be obtained when performing the SCA. The problem now is to do the above operations in a symbolic way, and hence solving the linear system symbolically as fast as possible. This problem is overcome and applied to solve the symbolic linear system that was converted into a sparse symbolic system in the wavelet basis. The system is then solved using sparse system solution technique, all in a symbolic fashion. This process reduces the time required to obtain the SCA output as will be shown later. The above steps were programmed using MATLAB, with the aid of some built-in modules.

D. The Wavelet Matrices and Sparsity

It can be seen from eq. (15) that the W matrix of dimension 4×4 will be as shown below:

$$W = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ c_3 & -c_2 & c_1 & -c_0 \\ c_2 & c_3 & c_0 & c_1 \\ c_1 & -c_0 & c_3 & -c_2 \end{bmatrix} \tag{25}$$

Where c_0, c_1, c_2, and c_3 are the DAUB4 filter coefficients as explained previously. The 4×4 matrix does not contain any zero.

Now, the W matrix of dimension 8×8 is as shown below:

$$W = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 \\ c_3 & -c_2 & c_1 & -c_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_0 & c_1 & c_2 & c_3 & 0 & 0 \\ 0 & 0 & c_3 & -c_2 & c_1 & -c_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_0 & c_1 & c_2 & c_3 \\ 0 & 0 & 0 & 0 & c_3 & -c_2 & c_1 & -c_0 \\ c_2 & c_3 & 0 & 0 & 0 & 0 & c_0 & c_1 \\ c_1 & -c_0 & 0 & 0 & 0 & 0 & c_3 & -c_2 \end{bmatrix} \tag{26}$$

Note the sparsity as the dimension of the matrix increases. The above matrix contains 64 elements, 32 of them are zeros (50% sparsity). The W matrix of dimension 16×16 contains even more zero entries and so on for higher order of W matrices. This will lead, when used to transform a linear system, to obtain a sparse system that makes its solution easier and faster. Table I shows a comparison between the size of the W matrices and the number of zeros included.

TABLE I
COMPARISON BETWEEN THE SIZE OF WAVELET MATRICES AND THE NUMBER OF ZEROS

Size of W Matrix	Total Number of Elements	Total Number of zeros	Sparsity ratio* (%)
4×4	16	None	0.00
8×8	64	32	50.00
16×16	256	192	75.00
32×32	1024	896	87.50
64×64	4096	3840	93.75
128×128	16384	15872	96.88

* Sparsity ratio= (Number of zeros)/(Number of elements)

It was found that the number of zeros in W matrix for DAUB filter is covered by the formula:

$$Z = D^2 - 4D \quad (27)$$

Where Z is the number of zeros and D×D is the dimension of the matrix (D = 4, 8, 16, ...).

IV. Simulating Results

This section presents examples of using the previously explained method that depends on the use of the DWT to solve the linear system matrix of circuits to be analyzed. The results are all compared to the conventional technique of numerical interpolation that uses FFT as a main transform. The comparison is based mainly on the computation overhead (amount of calculations) and hence the execution time.

The software required to test the proposed method is the MathWorks™ MATLAB ver. 7.5. For the purpose of fair comparison, two versions of the symbolic analysis program were written and tested. The first program that applies FFT in numerical interpolation is called SAUFFT (Symbolic Analyser Using Fast Fourier Transform), and the second program that applies DWT is called SAUDWT (Symbolic Analyser Using Discrete Wavelet Transform). The same circuits were used in both programs and the output obtained using a computer that has Intel Core Duo Processor operates on 1.6 GHz with 2 GB RAM.

A. Example 1

Consider the circuit shown in Fig.1. This circuit contains 8 symbolic variables which are C1, C2, C3, C4, gm1, gm2, gm3, and gm4, where the gm's are the transconductances of the Operational Transconductance Amplifier (OTA) devices. Being an active device, the OTA has been modelled using the nullator-norator equivalent circuit [16,17,18]. It is required to obtain the symbolic expression for the transfer function Vo/Vi. The description of this circuit was input to the program in a SPICE-like format. The description file of the circuit was fed to the two previously mentioned programs. The output of the two programs was the same but with different execution time. The output was as shown below:

THE NUMERATOR IS:

$$g_{m1}\, g_{m2}\, g_{m3}\, g_{m4}$$

THE DENOMINATOR IS:

$$C_1\, C_2\, C_3\, C_4\, s^4 + C_1\, C_2\, C_3\, g_{m4}\, s^3 + (g_{m3}\, g_{m2}\, C_1\, C_4 + C_1\, C_2\, g_{m4}\, g_{m3})\, s^2 + (g_{m3}\, g_{m2}\, g_{m1}\, C_4 + C_1\, g_{m4}\, g_{m3}\, g_{m2})\, s + g_{m3}\, g_{m1}\, g_{m2}\, g_{m4}$$

EXECUTION TIME OF SAUDWT: 1.2 sec.
EXECUTION TIME OF SAUFFT: 1.8 sec.

Note the difference in execution time between the two programs which is in favour of SAUDWT.

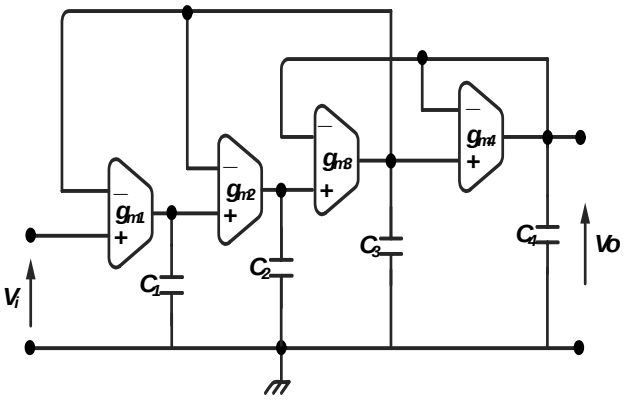

Fig. 1 Circuit of example 1

B. Example 2

Consider the circuit shown in Fig. 2. The circuit contains 11 symbolic variables with 4 operational amplifiers (OPAMPs) [19]. The output was as follows:

THE NUMERATOR IS:

$$-R_{10}\, R_7\, R_4\, R_5\, C_2\, R_2\, R_3\, C_1\, R_1\, s^2 - R_{10}\,(R_7\, R_4\, R_5\, C_2\, R_2\, R_3 - C_2\, R_2\, R_9\, R_1\, R_3\, R_5)\, s - R_{10}\, R_7\, R_4\, R_6\, R_1$$

THE DENOMINATOR IS:

$$R_7\, R_9\, R_4\, R_5\, C_2\, R_2\, R_3\, C_1\, R_1\, s^2 + R_7\, R_9\, R_4\, R_5\, C_2\, R_2\, R_3\, s + R_7\, R_9\, R_4\, R_6\, R_1$$

EXECUTION TIME OF SAUDWT: 1.6 sec.
EXECUTION TIME OF SAUFFT: 2.6 sec.

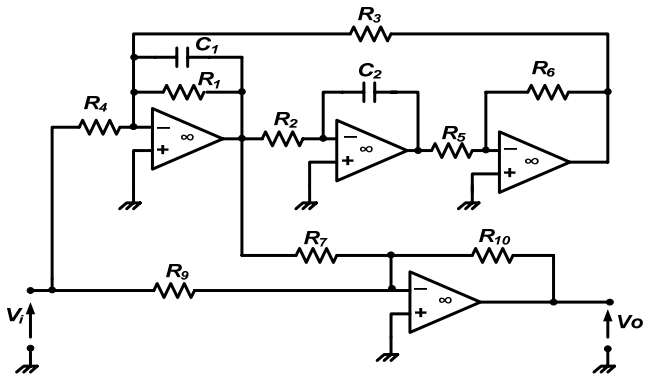

Fig. 2 Circuit of example 2

C. Example 3

Consider the RC ladder circuit shown in Fig. 3. It goes up to 60 sections. The circuit contains passive elements with 120 symbolic variables. This circuit is used to show the ability of the proposed method to tackle large circuits [20,21].

The output was obtained only using SAUWT program because SAUFFT program suffered from overflow due to the massive computations overhead. The execution time was *30.4 sec*., and the transfer function will not be shown in this context because it is too long.

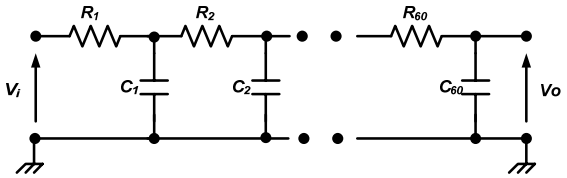

Fig. 3 Circuit of example 3

V. Performance Comparison Between The FFT And DWT In SCA

Fig. 4 shows an execution time comparison between the performance of the programs SAUFFT (that uses FFT) and SAUDWT (that uses DWT) when applied to SCA. It should be mentioned that the figure is obtained after using a certain set of circuits applied to both programs for the purpose of fair comparison. Of course, not only the number of symbolic elements affects the time required to analyze the circuit, but also the configuration of the circuit (that is, the number of nodes and branches). The figure shows the results up to 30 symbolic variables for the purpose of illustration.

It was found practically that SAUDWT performs better than SAUFFT in terms of the execution time and the ability to handle larger circuits as the number of symbolic variables increases. For small number of symbolic variables, however, the two programs perform almost the same with slight difference. Also, the program SAUDWT continues to provide the analysis with excellent time performance as the number of symbolic variables increases. This is not the case with SAUFFT, because as the number of symbolic variables increases, the required time increases rapidly and the analysis of larger circuits becomes impossible.

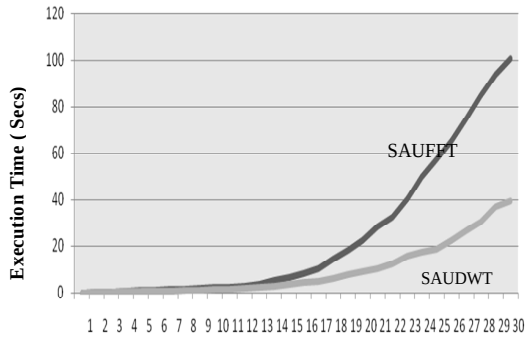

Fig. 4 Comparison between the execution times of programs SAUDWT and SAUFFT

VI. Concluding Comments

In this paper, a method for SCA was introduced. This method is based on the use of wavelet transform, namely the DWT, instead of the FFT for the numerical interpolation. Most of the usefulness of wavelets rests on the fact that wavelet transform can usefully be severely truncated, that is, turned into sparse expansions. The case of Fourier transform is different: FFT is ordinarily used without truncation, to compute fast convolutions, for example.

The subject of wavelet is developing fast and many questions remain to be answered, from these: what is the best choice of wavelet to use for a particular problem? Hence, by testing different wavelets, an optimum condition may be reached for the SCA application.

The ability of the wavelet transform to compress the data can be utilized highly in the area of SCA to facilitate the ability of analyzing large circuits without the massive computations overhead incurred in the old techniques. The proposed method and the program SAUDWT can be used with little or no modification to cope with large circuits (active and/or passive). Also, the CPU time and memory requirements are reduced drastically with regard to previous approaches.

References

[1] P. M. Lin, *Symbolic Network Analysis*, Elsevier Science Publishers, 1991, ch. 1.
[2] G. Gielen and W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits*, Kluwer Academic Publishers, 1991, ch. 3.
[3] H. Floberg, *Symbolic Analysis in Analog Integrated Circuit Design*, Kluwer Academic Publishers, 1997, ch.2.
[4] X. Wang and L. Hedrich, "Hierarchical symbolic analysis of analog circuits using two-port networks", *6th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing*, Cairo, Egypt, 2007.
[5] V. Fernández, P. Wambacq, G. Gielen, A. Rodríguez-Vázquez, and W. Sansen, "Symbolic analysis of large analog integrated circuits by approximation during expression generation", *Proc. Int. Symposium on Circuits and Systems, London*, pp.25-28, June 1994.
[6] G. Gielen, P. Wambaq, and W. Sansen, "Symbolic analysis methods and applications for analog circuits: a tutorial overview", *Proc. IEEE*, Vol. 82, No.2, February 1994.
[7] K. Singhal and J. Vlach, "Symbolic analysis of analog and digital circuits", *IEEE Transactions on Circuits and Systems*, Vol. CAS-24, No.11, November 1977.
[8] K. S. Yeung, "Symbolic Network Function Generation Via Discrete Fourier Transform", *IEEE Transactions on Circuits and Systems*, Vol. CAS-31, No. 2, 1984.
[9] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand-Reinhold, 1993.
[10] K. Singhal and J. Vlach, "Generation of immittance functions in symbolic form for lumped distributed active networks", *IEEE Transactions on Circuits and Systems*, Vol. CAS-21, No.1 January 1974.
[11] K. Yeung and F. Kumbi, "Symbolic matrix Inversion with application to electronic circuits", *IEEE Transactions on Circuits and Systems*, Vol. CAS-35, No.2, February 1988.
[12] S. Burrus, R. A. Gopinath, and H. Guo, *Introduction to Wavelets Transforms*, Prentice-Hall, 1998, ch. 4.
[13] E. Newland, *An Introduction to Random Vibrations, Spectral and Wavelet Analysis*, Longman Scientific and Technical, 1993, ch.7.
[14] H. P. William , T. V. William, A. T. Saul, and P. F. Brian, *Numerical Recipe, The Art of Scientific Computing*, Third Edition, Cambridge, 2007.
[15] I. Daubechies, "Orthonormal bases of compactly supported wavelets", *Comm. Pure Appl. Math.*, Vol. 41, 1988, pp. 906-966.
[16] P. Kumar and R. Senani, "Bibliography on nullors and their applications in circuit analysis, synthesis and design", *Analog Integrated Circuits and Processing*, 33, 65-76, Kluwer Academic Publishers, 2002.
[17] E. Tlelo-Cuautle, C. Sanchez-Lopez, and F. Sandoval-Ibarra, "Symbolic analysis: a formulation approach by manupulation data structures", *IEEE SCAS*, Vol. IV, pp.640-643, 2003.

[18] E. Tlelo-Cuautle, A. Quintanar-Ramos, G. Gutierrez-Perez, an M. Gonzalez de la Rosa," SIASCA: Interactive system for the symbolic analysis of analog circuits", *IEICE Electronic Express*, Vol. 1, No.1, pp.19-23, April 2004.
[19] Wai Kai Chen, *The Circuits and Filters Handbook*, CRC Press, 2003. ISBN: 0-8493-0912-3, ch.6.
[20] M. A. Al-Taee, F. M. Al-Naima, and B. Z. Al-Jewad, Optimized sparse storage mode for symbolic analysis of large networks, *Advances in Engineering Software*, Vol.38, No.2, Feb. 2007, pp.112-120.
[21] M. A. Al-Taee, F. M. Al-Naima, and B. Z. Al-Jewad, Symbolic analyzer for large lumped and distributed networks, Chapter 23 in the Book: *Symbolic – Numeric Computations*, Wang, Dongming, Zhi, Li-Hong (Eds.), Series: *Trends in Mathematics*, pp. 375-394, Birkhauser Verlag Basel, Switzerland, Feb 2007.

Fawzi Mohammed Munir Al-Naima (M'03) was born in Mosul, Iraq in 1948. He received both degrees of B.Sc. (First Class Honors), and Ph.D in Electrical Engineering from the University of Newcastle upon Tyne, UK in 1971 and 1976 respectively.

He worked as a Lecturer and Associate Professor in Al-Rasheed College of Engineering, Baghdad, Iraq from 1977 to 1989. He has been with the College of Engineering, Nahrain University, Baghdad, Iraq since 1989. He served as Head of Department of Electronics and Communication Engineering from 1992 to 2000, and Head of Department of Computer Engineering from 2000 to 2003, and then a Dean of the College of Engineering from 2003 to 2007 in the same university. He spent one Sabbatical year in Aleppo University, Syria from Oct 2006 to Sept 2007. Also, he worked as a Foreign Faculty Professor in the University of Engineering and Technology UET Taxila, Pakistan from Oct 2007 to Sept 2008.

Professor Al-Naima published more than 70 research papers in national and international journals and conferences. He is also the co-author of two chapters in two books published in 2002 and 2007 by international science publishers. His main research interests include computer aided design of large circuits, design of active filters, analog and digital signal processing and medical signal processing. Professor Al-Naima has been a Life Fellow Member of the IETE, India since 1999.