An Iterative Combinational Logic Synthesis Technique Using Spectral Information

Mitchell Aaron Thornton, V. S. S. Nair

Department of Computer Science and Engineering
Southern Methodist University

Abstract

The spectral information of a Boolean function provides data regarding the correlation between the input variables and the output of the function. This paper introduces a spectral based methodology for combinational logic synthesis using linear transforms. An analysis of the properties of the spectra obtained from these transforms is provided and a synthesis algorithm using spectral techniques is presented. This result is significant since it provides an algebraic method for including XOR gates in the synthesis process without resorting to manipulation of symbolic Boolean equations.

1 Introduction

The design and development of digital systems is becoming increasingly complex and automated. As fabrication technology evolves, a greater amount of circuitry per single chip results. This trend requires the designer to place more emphasis on optimization and efficiency during the chip design phase. Spectral based design techniques offer the advantage of providing for the design of a circuit of minimal size for a given functionality. These techniques are highly methodical and are ideal for incorporation into an automated environment. In the past literature, there have been a significant amount of results in the use of various spectral techniques for both binary-valued logic [1] [7] and multi-valued, or threshold logic [2] [3] [4]. Our approach differs from the previous works in that we allow the synthesized circuit to be composed of any of several types of sub-functions (i.e. AND, OR, combinations of AND/OR), whereas, the previous work focused on the realization of logic circuitry with predominately a single type of gate as the basic building block. It has been shown that the Rademacher-Walsh transform provides Boolean function output correlation measures with respect to various combinations of input variables added together via modulo-2 arithmetic [5]. This paper builds upon the fundamentals of spectral techniques and describes how they may be applied to the synthesis of general combinational logic circuits.

Typically, designers generate Boolean expressions at some point in the logic circuit design process regardless of the technique used. In fact, many logic design techniques are considered to be accomplished upon the realization of a Boolean function that is minimized in some sense. In reality, circuits are built from a schematic diagram or some other form of hardware description such as a wire-list or standard CAD tool file format. The synthesis technique discussed in this report does not require an intermediate Boolean function to be realized. Since there is no need for any symbolic manipulation, this technique is especially well suited for implementation as a computer program. Furthermore, the XOR gate is fully exploited as a potential candidate for incorporation into the design in addition to the AND and OR gates. Most design techniques require symbolic manipulation of Boolean functions to achieve inclusion of the XOR gate in the final circuit since numerical algebraic techniques are typically not well suited for this purpose. The algebraic methods presented here are rigorously developed and are very well suited for XOR type circuits.

The rest of the paper is organized as follows. Section 2 of this paper defines and describes the concept of the spectrum of a Boolean function. In this section, spectral properties that are important for the development and motivation for the synthesis technique are given. The synthesis method is developed and two examples are given in section 3. In section 4, a discussion of the implementation issues for the synthesis methodology is provided. Finally, the conclusions from this work are presented in section 5.

0-8186-4350-1/93 $3.00

2 Spectrum of a boolean function

First, the following terms are defined:

- **Output Vector of a Boolean Function:** A concatenation of all possible outputs of the function where all "0" values have been changed to a "1" and all "1" values have been changed to "-1".
- **Spectrum of a Boolean function:** A linear transformation of the output vector of the function. This transformed quantity is the "spectral vector" of the transformed function. The components of the the spectral vector are referred to as "spectral coefficients".
- **Transformation Matrix:** The matrix used to perform the linear mapping of a functions' output vector to its spectral vector.
- **Constituent Functions:** Boolean functions whose output vectors are used as the rows of the transformation matrix.

The transformations used in this approach are linear and they are conveniently visualized as vector-matrix products although the implementation may not necessarily be realized as a series of vector-matrix products. Multiplication and addition operations are performed over the field of integer numbers, not over the binary field. Using "1" and "-1" instead of "0" and "1" for the binary valued digits allows the zero-valued function outputs to accumulate in each spectral quantity.

Each spectral coefficient in the transformed output vector provides a measure of correlation between a particular constituent function and the original function, $F(x)$. This is the underlying principle that is exploited in the synthesis procedure described in this paper.

In general, any set of constituent functions may be used for this technique as long as they form a functionally complete set of operators for Boolean algebra. The transformation matrix must be full rank for convergence of this algorithm to be guaranteed. It has been shown that a transform using the Exclusive-OR operator as the primitive operation for the constituent functions results in the Rademacher-Walsh transform matrix. The Rademacher-Walsh transform matrix is orthogonal with well known properties [2]. In this paper, we construct transforms using the OR and AND operators in addition to the XOR to form the constituent functions. The resulting transforms are not necessarily orthogonal but our technique does not require the use of orthogonal transformations.

The notion of an output vector of a boolean function was described earlier. A qualitative discussion of the derivation of the transform matrices extends this notion such that the rows of the transformation matrix are viewed as output vectors of constituent functions. The constituent functions are generally simple functions using only a single operator (although they need not be). For instance, if it is desired to compute the correlation between a constituent function, $x * y$, and a function to be transformed, one row of the transformation matrix would consist of the output vector of the constituent function, $x * y$. The spectral coefficient resulting from the dot product of this row and the function output vector provides a measure of correlation between the overall function and the constituent function, $x * y$. In fact, a measure of correlation with any arbitrary constituent function may be computed in this manner. Each correlation measure (or spectral coefficient) contains the information of the exact number of matching outputs between the constituent function and the transformed function for a given common set of inputs. Before giving a precise relationship between the spectral coefficients and the number of matching outputs, the following parameters are defined.

1. n - the number of input variables of a Boolean function
2. N - all possible combinations of the input variables of the binary-valued Boolean function, $N = 2^n$
2. N_m - the number of matches between the outputs of a constituent function and the function to be synthesized for the same input values
3. N_{mm} - the number of mismatches between the outputs of a constituent function and the function to be synthesized
4. $S_F[F_c(x)]$ - the spectral coefficient of $F(x)$ that corresponds to the constituent function, $F_c(x)$
5. $S_F(1)$ - the spectral coefficient of $F(x)$ corresponding to $F_c(x) = 1$
6. $\{F_c(x)\}$ - a set of constituent functions
7. $\underline{S_F}[\{F_c(x)\}]$ - a vector of spectral values computed over the set of constituent functions, $\{F_c(x)\}$

Next, some properties of the spectral coefficients are derived and they will be used for the development of the synthesis technique given in this paper.

Lemma 1 *For a given function $F(x)$ and a given constituent function $F_c(x)$ the resulting spectral coefficient is given by:*

$$S_F[F_c(x)] = 2^n - 2N_{mm} = 2N_m - 2^n \qquad (1)$$

Where x is a n-order vector of binary valued inputs to both $F(x)$ and $F_c(x)$.

Lemma 2 *The following property of spectral coefficients holds:*

$$S_F[F_c(x)] = -S_F[\overline{F_c(x)}] \qquad (2)$$

The proofs for these lemmas are given in [8].

3 Synthesis method

The input required for this synthesis method consists of the truth table of the function, $F(x)$, the maximum number of inputs per gate, N_{inp}, and preferences of the types of gates, $\{G_i\}$, to be used. The two optional parameters, N_{inp} and the set $\{G_i\}$ are used to determine the set of constituent functions, $\{F_c(x)\}$ to be used in the formation of the transformation matrix. For the sake of generality, we will first assume that the optional parameters are not supplied, however, we will use them in the subsequent synthesis example.

The following list gives a detailed description of each synthesis step.

1. Convert the input truth table to 1's and -1's using a 1 to denote a logic"0" and a -1 to denote a logic "1".
2. Compute the transformation matrix using the constituent functions.
3. Compute the spectral coefficients via vector-matrix multiplication between the transformation matrix and the output vector of the function to be synthesized.
4. Choose the largest (in magnitude) spectral coefficient.
5. Realize the function $F_c(x)$ that corresponds to the chosen coefficient in step 4.
6. Compute the error function, $e(x) = F_c(x) * F(x)$ with respect to some operator, $*$.
7. If $e(x)$ indicates that there are w or fewer errors, go to step 8. Otherwise iterate on the synthesis by going to step 3 and use $e(x)$ as the next function to be synthesized.
8. Combine all the intermediate realizations of the various chosen $F_c(x)$ functions using the $*$ operator and directly realize the function $e(x)$ for the remaining w or fewer errors.

This technique can be used to generate two-level and multi-level tree-type circuits. For two-level realizations, each chosen $F_c(x)$ can be realized in the first stage of the circuit with one multi-input logic gate. The second stage consists of a single combination gate that uses the outputs of all of the chosen constituent functions as its' inputs. The circuits resulting from this synthesis technique are completely fan-out free (CFOF) and have the desirable property of requiring a set of test vectors equal to the number of circuit inputs to test all possible single stuck-at faults [6]. As discussed in [7], the use of spectral design techniques for logic synthesis is known for the ability to produce easily tested circuits. The diagram in Figure 1 indicates how the two-level circuit is constructed with each iteration.

If a multi-level circuit is desired, the same design procedure is used but the error function computation is performed slightly differently. The difference is that a new combination gate is used in each iteration. This also allows for changing the operator used to define the error function (i.e., the combination operator) at each iteration. The diagram in Figure 2 illustrates the design procedure for a multi-level circuit. Although it is apparent that the transformation matrix grows in size proportional to the size of the x vector, the space complexity of the matrix is $O(n^2)$ if the design is restricted to using only two-input gates [8].

The following theorem states the properties necessary to ensure the convergence of this synthesis algorithm.

Theorem 1 *Any given boolean function, $F(x)$, may be realized with the proposed synthesis technique if the transformation matrix used for the synthesis is of full rank.*

The proof for this theorem is given in [8]. Now some examples of this synthesis technique are given. In the first example, a two-level circuit will be designed and we will assume that there are no restrictions on the number of inputs per logic gate and on the type of logic gate used. The second example shows how a multi-level circuit can be realized with the constraints that only two-input gates should be used and that the gate type should be OR as much as possible.

Consider the realization of the following function:

$$F(x) = \overline{x}_1\overline{x}_3 + x_1\overline{x}_2x_3 + \overline{x}_1x_2 + x_2\overline{x}_3 \qquad (3)$$

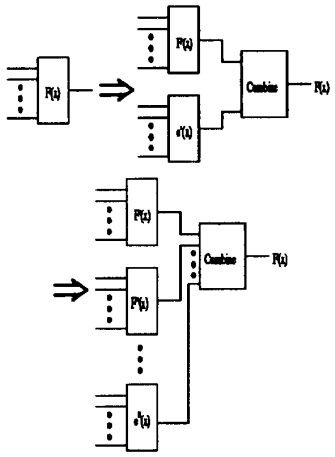

Figure 1: Diagram of Two-Level Synthesis Technique

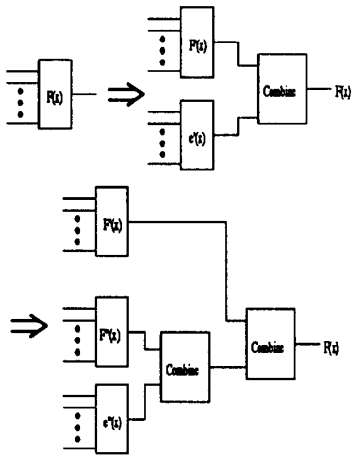

Figure 2: Diagram of Multi-Level Synthesis Technique

First, the function truth table is computed. Table 1 shows the contents of the computed truth table.

x_1	x_2	x_3	F
1	1	1	-1
1	1	-1	1
1	-1	1	-1
1	-1	-1	-1
-1	1	1	1
-1	1	-1	-1
-1	-1	1	-1
-1	-1	-1	1

Table 1: Truth Table of the Function

Next, the transformation matrix is computed in terms of OR, AND, and XOR operations since we have no constraints on gate types. Also, for each gate type we compute $F_c(x)$ functions for all possible combinations of two or more inputs since we have no restrictions on the maximum number of inputs per gate. The transformation matrix is computed as:

$$
\begin{array}{r}
1 \\ x_1 \\ x_2 \\ x_3 \\ x_1 \oplus x_2 \\ x_1 \oplus x_3 \\ x_2 \oplus x_3 \\ x_1 \oplus x_2 \oplus x_3 \\ x_1 + x_2 \\ x_1 + x_3 \\ x_2 + x_3 \\ x_1 + x_2 + x_3 \\ x_1 x_2 \\ x_1 x_3 \\ x_2 x_3 \\ x_1 x_2 x_3
\end{array}
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\
1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & -1 & 1 & -1 \\
1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & -1
\end{bmatrix}
$$

The resulting spectral coefficient vector is:

$$\underline{S}_F^T[\{F_c(x)\}] = [-2,-2,2,-2,2,-2,2,-6,2,$$
$$\text{-2,2,2,-2,-2,-2,-4] (4)}$$

Since the $F_c(x) = x_1 \oplus x_2 \oplus x_3$ has the largest magnitude spectral coefficient, it is chosen and the first portion of the circuit is realized with an exclusive-NOR as shown in Figure 3. The error function is computed with respect to an exclusive-OR operator since it is the most robust in terms of the possible operators available for providing the combining stage in the circuit.

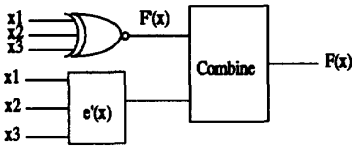

Figure 3: First Iteration of Two-Level Synthesis

This robustness is due to the fact that an XOR can be used to change a 0 to 1 error as well as a 1 to 0 error. The following list describes the properties that determine which gate type may be used as an error operator.

1. XOR : $x \oplus 1 = \overline{x}$, errors may be 1→0 or 0→1
2. AND : $x1 = x$ and $x0 = 0$, all errors must be 1→0
3. OR : $x + 1 = 1$ and $x + 0 = x$, all errors must be 0→1

Returning to the synthesis example, the following table is computed:

x_1	x_2	x_3	$F(x)$	$F'(x)$	$e(x)$
1	1	1	-1	-1	1
1	1	-1	1	1	1
1	-1	1	-1	1	-1
1	-1	-1	-1	-1	1
-1	1	1	1	1	1
-1	1	-1	-1	-1	1
-1	-1	1	-1	-1	1
-1	-1	-1	1	1	1

Table 2: Table of the Function and Error Function

Since there is only 1 disagreement between $F(x)$ and $F'(x)$, the terminal condition has been reached and the remaining term is realized directly. The complete circuit is given in Figure 4. The multi-level realization

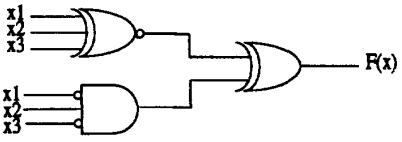

Figure 4: Final Circuit in Example 1

of the same circuit with the restriction that all gates are two-input and must be OR type gates as much as possible (inverters are also allowed) is given for the second example. The OR and NOT operator cannot be used to form a functionally complete set of operators for Boolean algebra. Therefore, at least one other type of gate is needed for the synthesis.

The transformation matrix is computed in terms of the Chow parameters and two input OR expressions only. The following spectrum results:

$$\underline{S_F^T}[\{F_c(x)\}] = [-2, -2, 2, -2, 2, -2, 2] \qquad (5)$$

In this case, all spectral coefficients have equal magnitudes. The constituent function $F_c(x) = x_1 + x_2$ is arbitrarily chosen as a starting point for the synthesis. It is also desirable to use an OR operation for the error function operator since this circuit is to be realized with predominately OR-type gates. By examining the truth table it is seen there are 0 to 1 and 1 to 0 discrepancies which restrict the error operator to be of type XOR. Table 3 contains the truth table in terms of the function to be realized and the error function.

x_1	x_2	x_3	$F(x)$	$e(x)$
1	1	1	-1	-1
1	1	-1	1	1
1	-1	1	-1	1
1	-1	-1	-1	1
-1	1	1	1	-1
-1	1	-1	-1	1
-1	-1	1	-1	1
-1	-1	-1	1	-1

Table 3: Table of the Function and Error Function

Next, the spectrum for $e(x)$ is computed, resulting in the following vector of spectral coefficients:

$$\underline{S_e^T}[\{F_c(x)\}] = [2, 2, -2, -4, -2, -2, -6] \qquad (6)$$

The constituent function corresponding to -6 is chosen since it is a maximum (in magnitude) which is $F_c(x) = x_2 + x_3$. There is a discrepancy in only one place in the truth table, hence we can use a NOR to directly realize this term. The resulting circuit is given in Figure 5. Note that the only real difference between

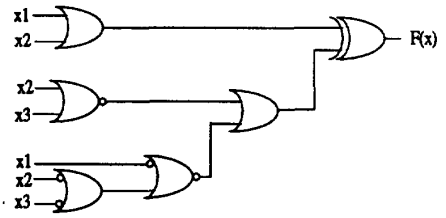

Figure 5: Final Circuit in Example 2

the two-level and multi-level synthesis techniques is in

the choice of new error operators at each iteration. This allows for greater flexibility when it is desired to use one gate type as much as possible since that type may be able to be used for the error operator in each iteration.

4 Implementation issues of the synthesis method

Although using repeated vector-matrix multiplications is a convenient way to analyze and understand spectral techniques for boolean function synthesis, it is not necessarily the most efficient way to implement these techniques. In particular, the computation of the Chow parameters may easily be performed by accumulating a single sum for each input of the function as the truth table is read in from a file and applying equation 1 to calculate the spectral value.

In general, all spectral coefficients may be computed efficiently as a simple accumulation of sums of the results of comparisons with transformation matrix rows and the current output vector being transformed.

In addition, it is quite easy to add extensions to this technique to produce output with desired properties. For instance, if only two-input gates are desired, all constituent functions in the transformation matrix are restricted to functions of two-inputs only. Also, by restricting the constituent function operators to a certain type allows the resulting circuit to contain only those operator (gate) types.

If optimization for circuit speed is desired, each spectral coefficient may be weighted by the inverse of the corresponding constituent function delay. This will result in the use of constituent functions with the least delay and highest correlation.

Existing standard cell logic libraries may also be used with this synthesis technique. All that is required is the output vector of each standard logic cell to be used as a row in the transformation matrix. This allows the synthesis technique presented here to be easily interfaced to existing design environments without the need for changing anything other than the "synthesis engine" itself.

5 Conclusions

We have presented a spectral based synthesis technique for general combinational logic circuits. Although this synthesis method has been developed within the framework of spectral analysis techniques, it may easily be considered to be a method of repeated correlation analysis. The spectral coefficients for a Boolean function have been defined and some of their properties have been derived. An iterative algorithm for the synthesis of combinational circuits has been developed and illustrated with examples. The algorithm has been proven to converge when the transformation matrix is of full rank. The proposed method can be implemented to produce circuits with optimizations such as gate count minimization and circuit delay minimization, or, to enforce restrictions such as the type of gate used and the number of inputs per gate. This synthesis methodology is significant since it allows for complete exploitation of the XOR gate as well as other gate types without resorting to symbolic manipulation of Boolean algebraic equations. Finally, some guidelines for the implementation of the logic circuit synthesis technique have been provided.

References

[1] Lloyd, A.M. Design of multiplexor universal-logic-module networks using spectral techniques, *IEE Proc.* **vol. 127, pt. E, no. 1** (1980).

[2] Edwards, C.R. The Application of the Rademacher-Walsh Transform to Boolean Function Classification and Threshold Logic Synthesis, *IEEE Trans. Comp.*, (1975).

[3] Damarla, T. Generalized Transforms for Multiple Valued Circuits and their Fault Detection *IEEE Trans. Comp.* **vol. C-41, no. 9** (1992).

[4] Picton, P.D. Realisation of multithreshold threshold logic networks using the Rademacher-Walsh transform, *IEE Proc.* **vol. 128, pt. E, no. 3** (1981).

[5] Hurst, S.L. The application of Chow parameters and Rademacher-Walsh matrices in the synthesis of binary functions, *Comput. J.* **vol. 16 no. 2** (1973).

[6] Pradham, D. K. (editor), **Fault-Tolerant Computing Theory and Techniques Volume I** Englewood Cliffs, NJ.:Prentice-Hall, (1986).

[7] Edwards, C.R. The design of easily tested circuits using mapping and spectral techniques, *Radio and Elec. Eng.* **vol. 47, no. 7** (1977).

[8] Thornton, M.A., Nair, V.S.S., Iterative Combinational Logic Synthesis Techniques Using Spectral Data, *Tech. Rep. 93-CSE-8*, SMU , (1993).