Stream Synthesis for Efficient Power Simulation Based on Spectral Transforms

Alberto Macii Enrico Macii Massimo Poncino Riccardo Scarsi
Politecnico di Torino
Dip. di Automatica e Informatica
Torino, ITALY 10129

Abstract

One way of minimizing the time required to perform simulation-based power estimation is that of reducing the length of the input trace to be fed to the simulator. Obviously, the use of a reduced stream may introduce some errors in the estimation results. The generation (or synthesis) of the short input sequence to be used for power simulation must then be done in such a way that the resulting error is minimized.

This paper introduces a new stream synthesis method whose peculiar feature is that of using spectral analysis techniques based on the discrete Fourier transform to determine a reduced sequence of vectors that enables to shorten the overall power simulation time at a very limited accuracy decrease.

The effectiveness of the proposed synthesis procedure is demonstrated by the results we have obtained on the Iscas'85 *combinational benchmarks for a variety of input streams characterized by different statistical and correlation properties.*

1 Introduction

Transistor-level simulation of typical input streams is by far the most accurate way of estimating the power dissipated by a CMOS digital circuit. The problem with this approach is that the traces to be simulated are usually extremely long; in addition, accurate low-level simulators are slow. Consequently, the time required to complete the estimation process can often become unacceptably long. In view of this, a wealth of research results have appeared recently on the development of techniques that enable a reduction of the total simulation time for both combinational and sequential circuits. Statistical sampling methods [1, 2, 3, 4, 5, 6] and multi-level schemes [7, 8] are just two examples of successful solutions.

One class of techniques that has lately received considerable attention is based on the idea of using probabilistic calculations to reduce the length of the original input stream without changing substantially some of its relevant statistical properties (e.g., signal and transition probabilities or spatio-temporal correlations among bits and across consecutive time frames) [9, 10, 11, 12, 13, 14, 15]. This with the goal of determining a new stream that can be simulated in a much shorter time at the price of a very limited average power estimation error.

Depending on the characteristics of the newly generated (short) set of binary input vectors, we distinguish between *compaction* and *synthesis* methods. (Notice that this is neither a standard classification nor a standard terminology, and it is introduced in this paper only for the sake of clarity.) Compaction procedures generate streams whose component patterns are all included in the original sequence (even though they may be sorted in a different way). Synthesis techniques, on the other hand, do not satisfy this constraint, that is, the streams they produce can contain patterns that do not appear in the original sequence.

In this paper, we propose a novel approach to stream synthesis that is based on the analysis of the spectral properties of the given input trace. Starting from the original stream, we first build an integer-valued function (called the *input switching function* hereafter) that expresses the number of switchings occurring between pairs of consecutive patterns as a function of time, and then we compute a discrete Fourier transform of such function. From the representation of the input switching function in the frequency domain we select some spectral coefficients, and then we go back to the time domain by computing the inverse transform of this restricted set of coefficients. The information provided by the new function in the time domain is exploited, together with the spatio-temporal and switching correlation information regarding the original stream, to synthesize a new stream, much shorter than the original one, whose simulation yields very accurate average power estimates.

The idea behind the proposed method is that of identifying the "components" of the original stream which are most relevant in characterizing the estimated value of the average power, isolating them, and mixing the information they provide with signal and transition probability measures, as well as spatio-temporal and transition correlation measures captured from the original input sequence. The stream synthesis procedure then exploits the results of the analysis phase to generate a reduced stream whose spectral characteristics, as well as statistical properties, are as close as possible to those of the original input trace.

We have performed an extensive experimentation on the complete set of the Iscas'85 [16] combinational benchmarks by applying our synthesis routine to a number of input traces of different nature. The data we have collected indicate how the addition of the spectral analysis step to a common correlation-based stream synthesis procedure considerably increases the robustness and the degree of automation of the procedure itself; in other words, it greatly helps in enhancing the tool's capabilities of properly and accurately handling streams with sensibly different statistical properties without requiring a modification or a customization of the selected correlation measures. It must be observed, however, that the peak performance (in terms of accuracy in the estimation of the average power consumption) obtained by some existing stream compaction/synthesis routines on input traces with specific statistical characteristics (less than 1% in some cases) are not always achievable using our method. Nevertheless, the data show an average estimation error ranging from 0.54% to 13.93%, depending on the considered input trace. It should be noted that the capability of characterizing the original input trace by means of spectral analysis constitutes the distinctive feature of the stream synthesis approach of this paper; to the best of our knowledge, in fact, no similar solution to the problem of compacting/synthesizing a sequence of vectors to be used for fast, yet accurate, average power estimation has been published in the literature. On the other hand, similarly to most of the existing solutions, our procedure has the desirable property of being *circuit independent*: It does not require the availability of the functional/structural description of the circuit to which the synthesized stream will be applied for simulation.

2 Background and Notation

In this section we recall some notions concerning Fourier transforms of discrete-time (or sampled) functions (Discrete Fourier Transforms) [17]. In the following, we assume the reader to be familiar with the definition and the fundamental properties of the Fourier transforms of continuous-time functions.

Let $x(t)$ be a periodic function of period P, and let T denote the sampling interval (i.e., the time interval between two consecutive samples) of $x(t)$; x can then be defined as a sequence of $N = \frac{P}{T}$ sampled values:

$$x(n) = \{x_0, \ldots, x_{N-1}\}$$

The periodicity assumption on function $x(t)$ is not mandatory; in fact, if $x(t)$ is only defined over a given time interval T_x, we can assume that $x(t)$ is indeed periodic of period T_x.

The usual Fourier integral that gives the transform $X(f)$ of function $x(t)$:

$$X(f) = \int_0^P x(t) e^{-j2\pi f t} dt$$

becomes, in the discrete-time domain:

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \tag{1}$$

where the transform $X(k)$ is a discrete function of a variable k which takes values in the range $\{0, \ldots, N-1\}$, and the discrete frequency points $\frac{k}{NT}$ replace the continuous frequency f. Equation 1 is called the *Discrete Fourier Transform* (DFT) of function $x(n)$.

The *Fourier spectrum* of $x(n)$ is thus defined by the series of complex coefficients $X(k)$, $k = 0, \ldots, N-1$, and function $x(n)$ is said to have a *line spectrum*.

The analysis of the right-hand side summation in Equation 1 shows that it is periodical, in n, of period N. It can be easily shown that:

$$X(k + mN) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi(k+mN)n/N} = X(k)$$

This in view of the fact that, for integer values of m:

$$e^{-j2\pi(k+mN)n/N} = e^{-j2\pi kn/N} \cdot e^{-j2\pi mn} = e^{-j2\pi kn/N}$$

Another useful property that stems from the time-domain sampling theorem is the *folding property*: A coefficient $X(N-k)$ is the complex conjugate of coefficient $X(k)$. In formula: $X(N-k) = X^*(k)$. This holds only for the spectrum of *real* (i.e., non complex) functions. In practice, the folding property tells us that only one half of the spectral coefficients need to be computed. The folding point $k = N/2$ is the discrete counterpart of the Nyquist frequency.

One final definition that will be used later in the paper is that of *inverse* DFT; when it is applied to the spectral coefficients, it reconstructs the corresponding sampled function. Similarly to the case of the inverse Fourier transform, the samples of the function are obtained as:

$$x(n) = \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N}$$

where $n = 0, \ldots, N-1$.

3 Stream Synthesis Methodology

In this section, we describe in detail a technique to synthesize short input traces to be used for power simulation in replacement of a given long stream of vectors. The method features the use of both spectral and correlation information collected on the given sequence to properly form a synthesized stream which guarantees a large speed-up in the simulation time at the price of a very low average power estimation error. We first provide a quick overview of our approach; then, we discuss in depth the most interesting steps of the stream generation process.

3.1 Overview

The high-level block diagram of the basic operations required by our stream synthesis methodology is depicted in Figure 1.

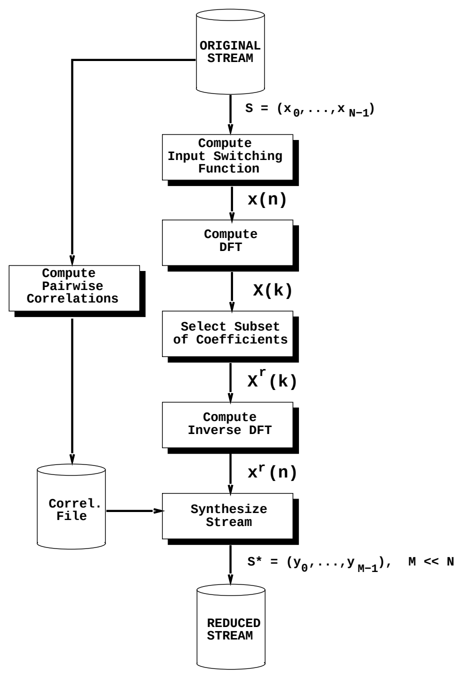

Figure 1: Block Diagram of the Stream Synthesis Methodology.

The entry point of the flow is a binary stream of N patterns, $S = (x_0, \ldots, x_{N-1})$, and the output is the synthesized stream $S^* = (y_0, \ldots, y_{M-1})$, with $M << N$. From stream S, the *input switching function* is first computed. This is an integer-valued function that represents the number of bit toggles between each pair of consecutive input patterns as a function of time. Time is discretized in correspondence of each pair of input patterns. In other words, the time point t is associated to the pair of consecutive patterns x_t and x_{t+1}. We will refer to the input switching function as $x(n)$ to emphasize its discrete nature.

Once $x(n)$ is determined, its DFT is calculated to get another sampled function $X(k)$ in the discrete frequency domain k ($X(k)$ is called the *spectrum* of $x(n)$, and the samples of $X(k)$ are the spectral coefficients of $x(n)$). Next, a subset of spectral coefficients (i.e., a subset of the samples of $X(k)$) is selected, so as to get a new sampled function, $X^r(k)$, in the frequency domain. Function $X^r(k)$, called *partial spectrum* of $x(n)$ in the sequel, is then transformed back to the time domain using the inverse DFT to get a new sampled function $x^r(n)$. If the choice of the subset of the spectral coefficients is carried out properly, $x^r(n)$ becomes an accurate approximation of $x(n)$ with a very limited number of samples.

The last step of the procedure consists of the actual synthesis of the reduced trace S^*. In a pre-processing phase, stream S is analyzed and the pairwise spatio-temporal and switching correlations are calculated. Then, the patterns in the new stream are generated in such a way that the input switching function of S^* is $x^r(n)$, and the correlations of S^* closely match those of S.

In the next sections we provide details on how the selection of the spectral coefficients and the synthesis of the reduced stream are carried out. On the other hand, we do not discuss the input switching function calculation phase, since it is quite straightforward. Also, we do not illustrate how the calculation of the pairwise correlation measures is done, since the subject has been extensively illustrated elsewhere (e.g., [18, 19, 9, 20]); we only report, in Section 3.3, some definitions that will help the comprehension of the stream generation procedure. Finally, we do not discuss how the DFT and its inverse are calculated, since standard numerical algorithms that efficiently perform these operations do exist [21]. However, it should be observed that, from the theoretical stand-point, the application of the DFT to a function is meaningful only if that function is periodic with a period P. In the case of $x(n)$, we can assume that this constraint is satisfied by thinking at $x(n)$ as a periodic function of period $P = N \cdot T$, where T is the sampling rate. In other words, instead of regarding $x(n)$ as a function defined over a limited time interval (namely, N clock cycles), we consider it as periodically repeating itself outside the interval $[0, P]$. Without loss of generality, in the following we assume $T = 1$, so that $P \equiv N$.

3.2 Selection of the Spectral Coefficients

We are given the representation $X(k)$ of $x(n)$ in the frequency domain. The samples of $X(k)$, usually called the spectral coefficients of $x(n)$, can be interpreted as the coefficients of a linear combination of sinusoids that approximates function $x(n)$. More specifically, $X(0)$ represents the DC-value, that is, the mean value of $x(n)$ over the period P of $x(n)$. The sinusoid for a given coefficient $k > 0$ has a period of $\frac{N}{k}$, and it is modulated according to the coefficient $X(k)$. Obviously, the multiples $m \cdot k$ of a given point k have a shorter period $\frac{N}{m \cdot k}$.

For our purposes, it is key to remember that the function in the time domain obtained by summing a fundamental sinusoid (i.e., the sinusoid at frequency k) to its harmonics (i.e., the sinusoids at frequencies $m \cdot k$) is still periodic with period $\frac{1}{k}$. frequency. We then have that, by selecting some value k_c as the fundamental frequency, and by suppressing all the remaining frequencies but the multiples (i.e., the harmonics) of k_c, we obtain a sampled function $X^r(k)$ in the frequency domain (i.e., the partial spectrum of $x(n)$) whose counterpart $x^r(n)$ in the time domain is a periodic function with period $P = \frac{N}{k_c}$.

The interpretation that can be given of the result above is that $x^r(n)$ constitutes the function with period smaller than P (i.e., a sub-multiple) that best approximates function $x(n)$ with respect to the quantity of information it carries. Then, it is possible to use *one period* of $x^r(n)$ to represent *one period* of $x(n)$.

In the context of stream synthesis, $x(n)$ represents the input switching function of trace S; therefore, function $x^r(n)$ we obtain from $X^r(k)$ through the inverse DFT provides us with the input switching function that best approximates that of S. In addition, since the period P of $x(n)$ is actually the length of the original trace, one period of $x^r(n)$ directly corresponds to a shorter stream. Consequently, the choice of the fundamental frequency k_c uniquely determines the *compaction ratio* (i.e., N/M), provided that the partial spectrum $X^r(k)$ is derived from $X(k)$ by removing all the coefficients that are not multiples of k_c (the DC-coefficient $X(0)$ is obviously kept in $X^r(k)$).

From the discussion above it results clear that the selection of the fundamental frequency k_c allows a flexible trade-off between accuracy in the estimated average power and compaction ratio. By picking a smaller value of k_c, we include in the partial spectrum a large number of frequencies, and therefore we guarantee a finer approximation of the original sampled function, at the price of a smaller compaction ratio. Conversely, by choosing a higher value of k_c we include in $X^r(k)$ fewer frequencies, thus privileging the generation of a shorter stream, at the cost of an increased estimation error. The limit, unrealistic case is obviously that of a desired compaction factor of N; in this case, only the DC-coefficient is considered.

In principle, after the desired value of the compaction ratio has been fixed, the choice of the fundamental frequency can be done arbitrarily, without any constraint. The theory of the Fourier transform, however, provides us with a criterion on how the value of k_c should be picked to maximize the effectiveness of the stream synthesis procedure. In fact, to guarantee the exact periodicity of $x^r(n)$, it is required that the largest frequency N is also a multiple of k_c, that is, $X^r(k)$ should contain the coefficient at frequency $k = \frac{N}{k_c}$. If this is not the case, the inverse transform of $X^r(k)$ yields a function $x^r_{approx}(n)$ in the time domain which is *quasi-periodic*. The use of the latter instead of $x^r(n)$ within the stream synthesis procedure introduces an error that decreases as the similarity between the two functions increases.

Example 1

Let us consider the plot of $x(n)$, derived from a stream of 128 input patterns, depicted on the left-hand side of Figure 2. The corresponding spectrum, $X(k)$, is reported on the right-hand side of the same figure, and it has a symmetric behavior around the folding point $k = N/2 = 64$.

If we select a fundamental frequency $k_c = 16$, we obtain the partial spectrum $X^r(k)$ shown on the left part of Figure 3. The application of the inverse transform yields the function $x^r(n)$ represented on the right part of Figure 3, which is periodic with period $P = \frac{128}{16} = 8$.

On the other hand, if the value $k_c = 25$ (which is not a submultiple of the period $P = 128$ of $x(n)$) is picked as the fundamental frequency, the plots for $X^r(k)$ and $x^r_{approx}(n)$ are those reported in Figure 4. The difference in periodicity between $x^r(n)$ and $x^r_{approx}(n)$ is apparent.

3.3 Stream Synthesis

The synthesis phase consists of the generation of the reduced stream S^* starting from the samples of function $x^r(n)$. The main constraint to be met during the synthesis step is that the reduced stream must yield an input switching function which is exactly $x^r(n)$. Furthermore, the transition probabilities and the correlations of S^* must tightly resemble to those of the original stream S.

Before going into the details of the synthesis procedure, we informally recall the definition of the two correlation measures (i.e., *spatio-temporal* and *switching* correlation) we have used as additional constraints during the generation of the binary patterns of S^*. For the derivation of the mathematical expressions of these measures, as well as for the description of a procedure to calculate them starting from a long stream of binary vectors, the interested reader can refer, for example, to [9, 20].

The *spatio-temporal* correlation between bits p and q of pattern i, indicated in the sequel as $\rho_{p,q}$, is the likelihood of correctly predicting the value of bit p of pattern i given the value of bit p of pattern $i-1$ and of bit q of pattern i.

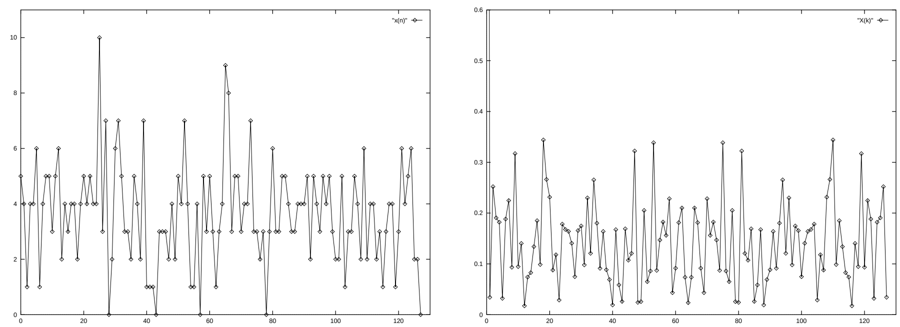

Figure 2: Input Switching Function $x(n)$ (Left) and its Spectrum $X(k)$ (Right).

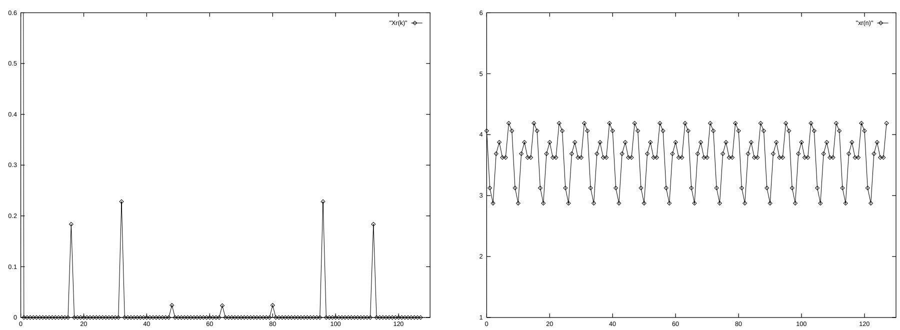

Figure 3: Partial Spectrum $X^r(k)$ (Left) and its Inverse DFT $x^r(n)$ (Right) for $k_c = 16$.

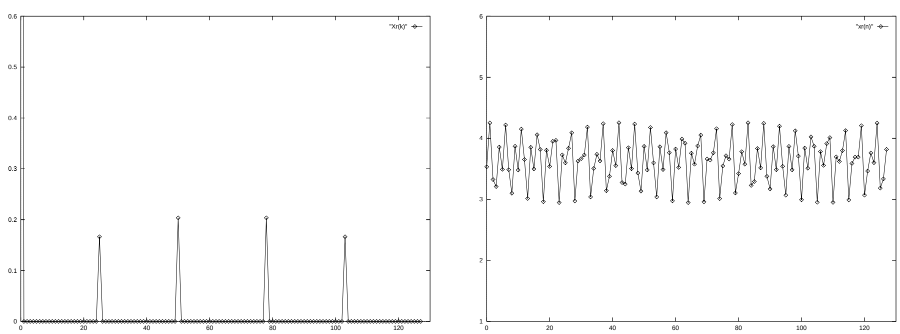

Figure 4: Partial Spectrum $X^r(k)$ (Left) and its Inverse DFT $x^r_{approx}(n)$ (Right) for $k_c = 25$.

The *switching correlation* between bits p and q of pattern i, indicated in the sequel as $\xi_{p,q}$, is the likelihood of correctly predicting a transition on bit p from pattern $i-1$ to pattern i given a transition on bit q from pattern $i-1$ to pattern i.

Since both correlation measures are important in modeling the power-determining characteristics of the original input trace, they must be considered simultaneously during the synthesis phase. We therefore define the *compound* correlation $\omega_{p,q}$ between bits p and q of pattern i as the convex combination:

$$\omega_{p,q} = c \cdot \rho_{p,q} + (1-c) \cdot \xi_{p,q}$$

where c is a real number between 0 and 1.

For efficiency reasons, the calculation of the compound correlations is performed concurrently with the construction of the input switching function $x(n)$, since both operations require to explicitly examine all the pairs of consecutive patterns in the original input trace. The results of the analysis are then stored into a matrix. It should be noticed that the decision of measuring correlations using a pairwise approximation is suggested by the fact that computing exact, word-level correlations may become computationally infeasible in the case of very long streams.

Given the definition of the compound pairwise correlation, $\omega_{p,q}$, we are now ready to discuss the algorithm for synthesizing the reduced input stream S^*, whose high-level pseudo-code is reported in Figure 5.

procedure `Synthesize_Stream` ($x^r(n)$, M, $\omega^S_{p,q}$, t^S_p, x_0) {
$y_0 = x_0$;
$S^* += y_0$;
$i = 1$;
while ($|S^*| < M$) {
  $y_i = y_{i-1}$;
  $y_i =$ `Generate_Vector` (y_i, $x^r(i)$, $\omega^S_{p,q}$, t^S_p);
  $S^* += y_i$;
  ($\omega^{S^*}_{p,q}$, $t^{S^*}_p$) $=$ `Update_Stats` (S^*, y_i);
  $i = i + 1$;
}
return S^* ;
}

Figure 5: The `Synthesize_Stream` Algorithm.

The input parameters of the procedure are the input switching function $x^r(n)$ of the target stream S^*, its length M, the correlation matrix $\omega^S_{p,q}$, the transition probability vector t^S_p, and the first vector x_0 of the original stream S. In the pseudo-code, superscript S is used to distinguish the measures that refer to S from those regarding the reduced stream S^*. The latter are iteratively updated by the algorithm any time a new pattern is added to stream S^*; this is required because the matching between the characteristics of S^* and S must be checked dynamically while the reduced stream is constructed.

The algorithm starts by choosing as the first pattern of the reduced stream, y_0, the first pattern of the original trace, x_0 (Line 1). Such pattern is added to S^* in Line 2. The procedure then enters the **while** loop of Line 3, in which the patterns to be added to the stream are iteratively selected until the desired length M of S^* is reached. At each iteration, the previously generated vector y_{i-1} is first duplicated (Line 4), and then used as the starting point for the generation of the new vector y_i through procedure `Generate_Vector` (Line 5). Such procedure constitutes the core of the synthesis algorithm; therefore, it will be described in details a little later in this section. After the newly generated pattern y_i is added to the stream (Line 6), the transition probability vector and the correlation matrix of S^* are updated in Line 7. Finally, the loop counter i is incremented in Line 8, and the vector generation process starts over.

For what concerns the generation of the new binary vectors (procedure `Generate_Vector` in Figure 5), we have devised the following strategy:

1. All the bits of the vector are initially marked as "unused";
2. The unused bit p whose transition probability deviates most from the value t^S_p (i.e., the bit p for which the difference $|t^S_p - t^{S^*}_p|$ is maximum) is selected, and its value is complemented.
3. The unused bit q whose compound correlation with respect to bit p deviates most from the value $\omega_{p,q}$ (i.e., the bit q for which the difference $|\omega_{p,q} - \omega^S_{p,q}|$ is maximum) is selected, and its value is complemented.
4. Bits p and q are then marked as "used".
5. If the number of transitions between pattern y_{i-1} and pattern y_i matches the expected number of switchings given by $x^r(i)$, the generation of y_i terminates. Otherwise, the process starts over from step 1.

Notice that we have chosen to modify only at most one pair of bits of y_i per iteration (i.e., bits p and q) because this seems to be more consistent with the type of correlation (i.e., pairwise) we use to drive the pattern generation process, Intuitively this strategy, unlike other possible options, should provide an easier way of re-constructing in stream S^* the statistical and correlation properties held by the original input trace S.

4 Experimental Results

In this section, we present experimental data regarding the use of the stream synthesis technique described in Section 3. The circuits on which the original and the reduced sequences of binary vector have been simulated are those included in the Iscas'85 combinational benchmark suite [16]. The switch-level netlists for the examples have been obtained by mapping their gate-level descriptions onto a 0.35μ CMOS industrial library, consisting of approximately 200 primitives. The average power values have been estimated using Irsim [22], and experiments have been run on a DEC AXP 1000/400 with 256 MB of RAM.

Since our objective is not only that of demonstrating the effectiveness (in terms of the estimation errors originated by the use of the synthesized stream instead of the original input sequence) of the proposed method, but also that of proving its robustness, we have applied it to a total of 10 distinct input traces, each of which is characterized by different statistical and correlation properties.

We have considered the following types of streams (each pattern is assumed to have K bits):

- *Type A*: Fully random patterns.
- *Type B*: Counter sequence restarting at a random number after a fixed number of patterns are generated [9].
- *Type C*: The transition probability of the most significant bit is set to $\frac{1}{k+1}$; moving towards the least significant bits, this value gracefully increases up to $\frac{k}{k+1}$.
- *Type D*: The transition probability of the most significant bit is set to $\frac{k}{k+1}$; moving towards the least significant bits, this value gracefully decreases down to $\frac{1}{k+1}$.
- *Type E*: The most significant $K/2$ bits of the patterns realize a counter sequence, the remaining bits are random.
- *Type F*: The most significant $K/2$ bits of the patterns are random, the remaining bits realize a counter sequence.
- *Type G*: Waveform with stair-case behavior [8].
- *Type H*: The transition probability of the bits follows a normal distribution centered on bit $K/2$.
- *Type I*: The transition probability of the most significant $K/2$ bits is high (0.7), that of the remaining bits is low (0.3).
- *Type J*: The transition probability of the most significant $K/2$ bits is low (0.3), that of the remaining bits is high (0.7).

We have considered original input streams consisting of 32,768 patterns. The choice of the number of vectors is dictated by the fact that most DFT algorithms require, for computational efficiency, a number of sampled points that is a power of two. This fact, however, does not constitutes a limitation of our method. A compaction ratio of 32X has been constantly used for all the experiments, thus constraining the synthesized streams to be made of a total of 1024 binary vectors.

Tables 1 and 2 collect the results we have obtained. For each type of input stream, we report the values of the average power, in mW, obtained by simulating both the original (column *Orig.*) and the reduced trace (column *Red.*). The percentage of estimation error is also reported (column *Err.*).

The experimental data are very satisfactory, since the average estimation error ranges from 0.54% to 13.93%, depending on the type of input stream used. The only case in which the average error has gone beyond 7% corresponds to the streams of *Type G*. To some extent, this situation was expected, since the statistical and correlation properties of such traces exhibit sensible variations over time; in addition, also the average power consumption is heavily time-dependent; consequently, reducing the number of patterns that must be simulated may have a sizable impact on the overall quality of the estimation. Instead of decreasing the length of the stream, a better way for quickly, yet accurately, tracking the power dissipated by sequences of this type consists of using multi-level simulation techniques such as those described in [8].

Circ.	Type A			Type B			Type C			Type D			Type E		
	Orig.	Red.	Err.	Orig.	Red.	Err.	Orig.	Red.	Err.	Orig.	Red.	Err.	Orig.	Red.	Err.
c432	0.733	0.699	4.60	0.208	0.228	9.61	0.680	0.668	1.76	0.698	0.702	0.57	0.563	0.568	0.88
c499	1.632	1.643	0.67	0.346	0.368	6.35	1.482	1.488	0.40	1.630	1.638	0.49	1.308	1.333	1.91
c880	1.246	1.243	0.24	0.458	0.424	7.42	1.322	1.330	0.60	1.058	1.060	0.18	0.965	0.951	1.45
c1355	1.632	1.642	0.61	0.346	0.368	6.35	1.482	1.488	0.40	1.630	1.638	0.49	1.308	1.333	1.91
c1908	1.928	1.898	1.55	0.761	0.814	6.96	1.727	1.747	1.15	1.762	1.787	1.41	1.572	1.616	2.79
c2670	3.128	3.211	2.65	0.823	0.777	5.58	3.227	3.233	0.18	2.604	2.598	0.23	2.119	2.133	0.66
c3540	5.177	5.338	3.10	2.826	3.061	8.31	3.456	3.409	1.35	6.155	6.166	0.17	4.488	4.995	11.29
c5315	7.947	7.909	0.47	1.137	1.138	0.08	9.153	9.170	0.18	5.895	5.896	0.01	5.755	5.814	1.02
c6288	11.820	12.064	2.05	1.611	1.725	7.07	12.659	12.661	0.01	11.960	12.175	1.80	9.916	10.094	1.80
c7552	13.660	12.940	5.27	2.326	2.543	9.32	10.897	10.910	0.11	15.738	15.727	0.07	9.717	9.652	0.67
Avg.			2.12			6.70			0.61			0.54			2.43

Table 1: Power Estimation Results for the Iscas'85 Combinational Circuits (Part I).

Circ.	Type F			Type G			Type H			Type I			Type J		
	Orig.	Red.	Err.	Orig.	Red.	Err.	Orig.	Red.	Err.	Orig.	Red.	Err.	Orig.	Red.	Err.
c432	0.549	0.552	0.54	0.612	0.688	12.40	0.697	0.700	0.43	0.676	0.696	2.95	0.718	0.736	2.50
c499	1.367	1.393	1.90	1.334	1.550	16.19	1.648	1.659	0.66	1.518	1.535	1.11	1.631	1.636	0.30
c880	0.847	0.790	6.73	1.199	1.332	11.10	1.136	1.124	1.05	1.394	1.364	2.15	1.080	1.130	4.63
c1355	1.367	1.393	1.90	1.334	1.550	16.19	1.648	1.659	0.66	1.518	1.535	1.11	1.631	1.636	0.30
c1908	1.629	1.646	1.04	1.634	1.858	13.70	1.793	1.839	2.56	1.587	1.548	2.45	1.897	1.889	0.42
c2670	2.124	2.143	0.89	2.567	2.919	13.71	2.325	2.334	0.38	3.182	3.215	1.03	2.990	3.045	1.83
c3540	3.632	3.702	1.92	4.564	5.215	14.26	4.397	4.371	0.59	3.919	3.812	2.73	5.805	5.789	0.27
c5315	5.762	5.892	2.25	7.844	8.594	9.56	7.921	7.885	0.45	9.451	9.445	0.60	6.182	6.050	2.13
c6288	9.941	10.384	4.45	11.270	12.095	7.32	12.607	12.570	0.23	12.810	12.970	1.24	12.010	12.030	0.16
c7552	9.575	9.684	1.13	10.580	7.940	24.90	8.275	8.292	0.20	11.750	11.690	0.51	14.563	14.561	0.01
Avg.			2.27			13.93			0.72			1.58			1.25

Table 2: Power Estimation Results for the Iscas'85 Combinational Circuits (Part II).

5 Conclusions

In this paper, we have proposed a novel technique to synthesize short streams of patterns that can be used for power simulation instead of the long input traces usually determined by the designers through architectural, behavioral or system-level simulation. The method features the use of spectral information (in addition to the usual correlation measures) collected on the given sequence to properly form a reduced stream which guarantees a large speed-up in the simulation time at the price of a very low average power estimation error.
The main advantage of the method is its robustness, that is, its capability of generating "good" reduced streams for a large variety of original input traces (i.e., streams characterized by different statistical and correlation properties), as demonstrated by the large set of experimental results we have presented.

Acknowledgments

We wish to thank the people from SGS-Thomson in Agrate Brianza for interesting discussions on fast power simulation.

References

[1] R. Burch, F. Najm, P. Yang, T. Trick, "A Monte-Carlo Approach for Power Estimation," IEEE Trans. on VLSI Systems, Vol. 1, No. 1, pp. 63-71, Jan. 1993.

[2] M. G. Xakellis, F. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," DAC-31, pp. 728-733, Jun. 1994.

[3] F. Najm, S. Goel, I. Hajj, "Power Estimation in Sequential Circuits," DAC-32, pp. 635-640, Jun. 1995.

[4] T-L. Chou, K. Roy, "Statistical Estimation of Sequential Circuit Activity," ICCAD-95, pp. 34-37, Nov. 1995.

[5] C-S. Ding, C-T. Hsieh, Q. Wu, M. Pedram, "Stratified Random Sampling for Power Estimation," ICCAD-96, pp. 577-582, Nov. 1996.

[6] V. Saxena, F. Najm, I. Hajj, "Monte-Carlo Approach for Power Estimation in Sequential Circuits," EDTC-97, pp. 416-420, Mar. 1997.

[7] S-Y. Huang, K-T. Cheng, K-C. Chen M. T-C. Lee, "A Novel Methodology for Transistor-Level Power Estimation," ISLPED-96, pp. 67-72, Aug. 1996.

[8] L. Benini, G. De Micheli, E. Macii, M. Poncino, R. Scarsi, "Fast Power Estimation for Deterministic Input Streams", ICCAD-97, pp. 494-501, Nov. 1997.

[9] C-Y. Tsui, R. Marculescu, D. Marculescu, M. Pedram, "Improving the Efficiency of Power Simulators by Input Vector Compaction," DAC-33, pp. 165-168, Jun. 1996.

[10] D. Marculescu, R. Marculescu, M. Pedram, "Stochastic Sequential Machine Synthesis Targeting Constrained Sequence Generation," DAC-33, pp. 696-701, Jun. 1996.

[11] R. Marculescu, D. Marculescu, M. Pedram, "Adaptive Models for Input Data Compaction for Power Simulators," ASPDAC-2, pp. 391-396, Jan. 1997.

[12] D. Marculescu, R. Marculescu, M. Pedram, "Sequence Compaction for Probabilistic Analysis of Finite State Machines," DAC-34, pp. 1-15, Jun. 1997.

[13] R. Marculescu, D. Marculescu, M. Pedram, "Power Estimation Using Hierarchical Markov Models," DAC-34, pp. 570-575, Jun. 1997.

[14] R. Marculescu, D. Marculescu, M. Pedram, "Composite Sequence Compaction for Finite State Machines Using Block Entropy and Higher-Order Markov Models," ISLPED-97, pp. 190-195, Aug. 1997.

[15] J. Kozhaya, F. Najm, "Accurate Power Estimation for Large Sequential Circuits," ICCAD-97, pp. 488-493, Nov. 1997.

[16] F. Brglez, H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," ISCAS-85, pp. 785-794, Jun. 1985.

[17] D. F. Elliot, K. R. Rao, Fast Transform: Algorithms, Analyses, Application, Academic Press, 1982.

[18] R. Marculescu, D. Marculescu, M. Pedram, "Switching Activity Analysis Considering Spatio-Temporal Correlations," ICCAD-94, pp. 294-299, Nov. 1994.

[19] R. Marculescu, D. Marculescu, M. Pedram, "Efficient Power Estimation for Highly Correlated Input Streams," DAC-32, pp. 628-634, Jun. 1995.

[20] L. Benini, G. De Micheli, E. Macii, M. Poncino, S. Quer, "System-Level Power Optimization of Special Purpose Applications: The Beach Solution", ISLPED-97, pp. 24-29, Aug. 1997.

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in C, Cambridge University Press, 1987.

[22] A. Salz, M. Horowitz, "IRSIM: An Incremental MOS Switch-Level Simulator," DAC-26, pp. 173-178, Jun. 1989.