Automated Synthesis of Large Phase Shifters for BIST

Janusz Rajski Nagesh Tamarapalli

Mentor Graphics Corporation
8005 S. W. Boeckman Road
Wilsonville, OR 97070, USA
e-mail: {janusz_rajski,
nagesh_tamarapalli}@mentorg.com

Jerzy Tyszer

Poznan University of Technology
ul. Piotrowo 3a
60-965 Poznan, Poland
e-mail: tyszer@et.put.poznan.pl

1. Introduction

Variety of generation schemes have been proposed in the open literature and are used in industrial practice to produce different types of test vectors that can be employed in BIST applications [4]. These schemes attempt to meet three basic requirements: high fault coverage, short test application time, and ease of incorporating necessary hardware onto a chip. The existing solutions include pseudo-exhaustive and pseudo-random techniques as well as more complex mixed-mode generation scenarios. All of them target a complete fault coverage, trying at the same time not to impose major circuit modifications and performance degradation. An advantage of the pseudo-random testing stems from its potential for very simple hardware and small design efforts required to implement test generation means. By far the most popular device in use as a source of pseudo-random test sequences is a linear feedback shift register (LFSR). However, as most design for testability schemes employ multiple scan chains to guarantee acceptable test application time, LFSRs are usually used to build two-dimensional test pattern generators feeding multiplicity of scan chains in parallel. In such environment, the resultant fault coverage is often unsatisfactory due to structural dependencies introduced by the test generator. If the scan paths are fed directly from adjacent bits of the LFSR, then this very close proximity will cause neighboring scan chains to contain test patterns which are highly correlated. This phenomenon can adversely affect fault coverage, as the patterns seen by the circuit under test (CUT) will not be pseudo-random. In order to alleviate this problem, an extra circuitry is placed between the LFSR and the serial inputs of the scan chains, and this part of the generator is referred to as a *phase shifter* [3].

A typical phase shifter consists of exclusive-or (XOR) trees used to avoid shifted versions of the same test data in various scan paths. Every XOR tree corresponds to a linear combination of LFSR stage outputs that results in the desired phaseshift for a given scan chain driven by this particular tree. In practice, phaseshift circuits are designed according to different principles. In *constant-phase shifters* [3], an interchannel displacement for each scan path is specified prior to the actual synthesis process. The basic deficiency of this approach is an unpredictable complexity of the resultant circuit which may feature, even after a decomposition and factorization process, a significant number of XOR gates. In *variable-phase shifters* [1], the design process starts with a prespecified structure of a phase shifter and subsequently determines the resultant channel phaseshifts. It allows one to control the amount of hardware involved, but it might be intricate to determine the actual phaseshifts [2]. Furthermore, the method used to design the variable-phase shifters is inherently an "ad hoc technique" and becomes impractical for the circuits with a large number of scan chains.

A first systematic design technique of phase shifters has been proposed in [8]. Using a concept of LFSR duality it relates the logical states of LFSRs and circuits spacing their inputs to each of the output channels. Consequently, the method generates a phase shifter network satisfying criteria of channel separation and circuit complexity by taking advantage of simple logic simulation of the LFSRs. Since efficiency of this technique depends on the size of the LFSR, the resulting phase shifter synthesis time may become unacceptable for large LFSRs, especially if a primary design objective is to reduce necessary hardware. In this presentation, we propose a novel design technique that overcomes this drawback while maintaining all advantages of the LFSR duality-based synthesis approach. Rather than seeking linear combinations for each channel satisfying some criteria, the new method selects successive linear combinations of LFSR stages in a lexicographic order and subsequently verifies whether they can be accepted based on channel separation criterion. We will present experimental results obtained for numerous practical cases in order to demonstrate high efficiency of the proposed technique when synthesizing phase shifter networks capable of driving a large number of scan chains with minimal area overhead and small delay time.

2. Problem formulation

Given an LFSR with a characteristic primitive polynomial $h(x)$ of degree n, find a linear combination of the LFSR stage outputs such that the resulting sequence, obtained as an exclusive-or of these stages, will be shifted by a pre-specified number of bits with respect to a reference channel. It was shown [4] that in order to calculate tap points for a channel spaced q shifts down the m-sequence from the channel represented by the LFSR jth stage, it is necessary to determine M^q, where M is an LFSR's transition matrix, and then multiply it by an appropriate selection vector, i.e., a vector that

features a single 1 in the jth column. The resulting vector will feature a 1 at each position that has to be added (modulo 2) to obtain the desired m-sequence. In particular, if the reference channel is fed by the LFSR leftmost output, the corresponding XOR tap positions can be found in the first column of M^q.

Although this result provides a basic link between transition matrices of LFSRs and architecture of phase shifters, its direct application is of limited use. This is because of matrix operations that may significantly impact the processing time. This limitation is overcome by the method introduced in [8,9]. It completely eliminates matrix multiplications and replaces them with logic simulation of LFSRs. This new technique exploits the concept of duality of LFSRs [5]. Given a type I LFSR, the type II LFSR obtained by reversing the direction of all feedback taps but the rightmost one is called its dual. Similarly, given an LFSR of type II, a dual LFSR of type I can be derived by reversing all the feedback taps except the rightmost one. It has been shown [8] that the contents of the first column of M after k multiplications will also be seen in the first row of D – a transition matrix of a dual LFSR - after 2^{n-1-k} multiplications of this matrix by itself. Since this row of D can be regarded as the state of the corresponding LFSR after applying 2^{n-1-k} clock cycles (provided its initial value was 10...0), the costly raising of M to successive powers can be replaced with a simple logic simulation of a type II LFSR. Algorithm 1 provides the basic operations that should be carried out in order to determine which outputs of the LFSR have to be XOR-ed to produce a desired string of bits that will be spaced k shifts down the reference m-sequence:

ALGORITHM 1: SYNTHESIS OF PHASE SHIFTERS
1. Determine a dual LFSR.
2. Initialize the dual LFSR to the value of 10...00.
3. Simulate the dual LFSR for 2^{n-1-k} clock cycles.
4. Read the resulting content of the dual LFSR: the locations of 1s will point out positions that should be included in a phase shifter structure to obtain a sequence shifted by k bits.

Simulation of the dual LFSR can be terminated after only k cycles. The state of the dual LFSR will then define the tap points for a channel moved k bits up the reference m-sequence, i.e., the m-sequence originating from the leftmost stage of the LFSR will be spaced k shifts down the new m-sequence. In the remaining we will use this convention.

3. New synthesis algorithm

For large LFSRs driving a large number of scan chains, a direct application of Algorithm 1 may lead to unacceptable processing time. As reported in [8], it takes approximately 20 minutes to obtain a phase shifter network fed by 32-bit LFSR, driving 256 scan chains with the minimum interchannel separation of 1024 bits and maximum 3 XOR taps per channel. This behavior can be attributed to the fact that binary combinations (treated here as states of a dual LFSR) having at most three ones are relatively seldom in the m-sequence produced by a 32-bit LFSR. Thus, a lot of states have to be scanned before accepting 256 combinations that will guarantee a requested minimum separation of 1024 bits.

As Algorithm 1 becomes computationally expensive with the increasing size of LFSRs while trying to maintain an XOR taps count low, an alternative approach can be employed that combines the accuracy of simulation-based technique with a new XOR taps selection rules. The essence of the new method is to generate successive XOR taps combinations in a systematic manner, for instance by recreating their lexicographic order, and then to check if a given combination can be chosen based on interchannel separation criterion. Since hardware complexity is one of the primary synthesis targets, it is justified to start the XOR tap generation process from the simplest combinations, i.e., those comprised of only two LFSR stages. The following procedure assumes that an auxiliary vector A it uses has dimension equal to the LFSR size and is employed for storing successive XOR taps. There are also global variables p and k such that the latter one records the current number of XOR taps which are used per channel.

ALGORITHM 2: XOR TAPS GENERATION
if A[k] = *LFSR size* **then** $p :=$ $p - 1$ **else** $p := k$
if $p >= 1$ **then**
    for $i := k$ **to** p **do** A[i] := A[p] + $i - p + 1$
else
    $k := k + 1$
    if k = *LFSR size* **then** this is the last combination
    $p := k$
    for $i := 1$ **to** k **do** A[i] := i

Assuming that initially A[1] = A[2] = 1, $k = 2$ and $p = 2$, the algorithm fills in the table A in a manner that corresponds to producing k-element sequences chosen from an n-element set using lexicographic order. In other words, successive calls of Algorithm 2 for 4-bit LFSR will yield the following XOR taps combinations: (1,2), (1,3), (1,4), (2,3), (2,4), (3,4), (1,2,3), (1,2,4), (1,3,4), (2,3,4), (1,2,3,4).

The next step is to ensure that a sequence produced by a linear combination of newly generated XOR taps does not overlap with already existing channels with respect to a requested phaseshift. Consider two sequences, F and B, comprising d successive states of a dual LFSR, such that F starts with the binary combination β representing the new XOR taps and runs forward, whereas B starts with the same pattern but runs backward. In order to guarantee the minimum channel separation d, the following requirement must be met: throughout sequences F and B there are no binary vectors corresponding to XOR taps combinations accepted in the previous generation steps. This condition implies that from state β onwards, for at least next d steps, there is no beginning of another sequence which is generated by a linear combination of XOR taps already included into a phase shifter network. It also demands for a similar absence of already selected combinations among, at least, d successive states preceding state β in the state trajectory of a respective dual LFSR.

We will now describe a simulation process that addresses both of these situations. Clearly, a forward simulation of LFSRs of both types can be easily accomplished by means of simple software modules commonly available in logic simulation tools. In order to determine states preceding a given binary vector, it suffices to observe that the inverse matrix M^{-1} runs the corresponding LFSR backward. Every combination generated during forward and backward simulations of dual LFSRs becomes the subject of searching for the same element in a dictionary of already accepted XOR taps, provided the number of ones this combination features does not exceed the actual value of k in Algorithm 2. Note that the last condition is extremely important as it may significantly reduce the number of search operations by means of preliminary inspection of new patterns. Indeed, vectors having more ones than k cannot be inserted into the dictionary, and thus the search does not have to be invoked. Several data structures can serve here as a dictionary of existing XOR taps combinations. In experiments we have performed, a hash table has been used with open addressing system employing double hashing to compute the probe sequences.

Having detailed the basics of XOR taps generation and validation, the main phase shifter synthesis algorithm can now be outlined. The following pseudocode assumes that routines *LFSR.1.back*, *LFSR.2.back*, *LFSR.1.forward* and *LFSR.2.forward* are used for backward and forward simulation of type I and type II LFSRs, respectively. The procedure uses an auxiliary variable *Ones* that records the number of ones occurring in the current states of simulated LFSRs.

ALGORITHM 3: FAST SYNTHESIS OF PHASE SHIFTERS

```
channels := 0
while channels < requested outputs do
    generate XOR taps set using Algorithm 2
    rejected := false

    initialize dual LFSR
    i := 0
    while i < separation and not rejected do
        if Type = 2 then LFSR.1.back else LFSR.2.back
        if Ones <= k and already generated then rejected := true
        i := i + 1

    initialize dual LFSR
    i := 0
    while i < separation and not rejected do
        if Type = 2 then LFSR.1.forward else LFSR.2.forward
        if Ones <= k and already generated then rejected := true
        i := i + 1

    if not rejected then
        insert XOR taps set into the dictionary
        channels := channels + 1
```

The code for Algorithm 3 consists of four parts. In each step of the outer **while** loop the procedure first generates a new XOR taps combination and subsequently checks if this combination can be accepted. If this is the case, the new combination is added to the dictionary, and the number of generated channels is updated. The actual validation is carried out by two internal **while** loops. After initialization of LFSRs, these loops advance dual LFSRs, in each step, by one clock cycle backward and forward, respectively, and then cause the XOR taps combination to be rejected if the current content of either LFSR matches any item kept in the dictionary. Both loops terminate successfully if counter i reaches a value of requested interchannel separation. Note that setting variable *rejected* immediately invokes a next generation of XOR taps.

As can be seen, the time complexity of Algorithm 3 is independent of the LFSR size. The proposed synthesis procedure runs in time proportional to $2shc$, where s is the minimum channel separation, c is the number of generated XOR taps

combinations, and h represents time necessary to complete search and insert operations in the hash table. Since under reasonable assumptions, the expected time to search for an element in a hash table can be O(1), the efficiency of Algorithm 3 is mainly determined by the minimum channel separation and the number of channels generated during the synthesis process, including those which were rejected. The impact of this factor can be reduced by modifying Algorithm 2 in such a way that it avoids some combinations, for instance those which are obvious shifted versions of taps selected in the previous steps (such combinations have a negligible chance to be accepted because of very small interchannel separation).

4. Experimental results and conclusions

It can now be seen that effectiveness and simplicity of Algorithms 2 and 3 allow automated synthesis of phase shifter networks with required properties regarding minimum separation between channels and minimal area overhead. To support this conjecture, synthesis of various large phase shifters has been performed on 75MHz Sun Ultra Sparc workstation. The obtained results clearly lead to the following observations.

Observation 1: The number of 2-way equivalent XOR gates needed to implement a phase shifter network is virtually the same as the number of output channels, i.e., the average number of 2-way XOR gates per channel is equal to one.

Observation 2: Given a characteristic primitive polynomial, a requested number of output channels and a minimum separation between them, the CPU time needed to complete synthesis of a resulting phase shifter is evidently shorter when a type I LFSR is used rather than an LFSR of type II.

Observation 3: Given a characteristic primitive polynomial, a requested number of output channels and a minimum separation between them, the resulting complexity of a phase shifter is visibly larger if the type I LFSRs are used.

Clearly, all numerical results support the conjecture that it is possible to obtain a very large and fast phase shifters with guaranteed properties regarding channel separation, minimal area overhead (around one XOR gate per scan chain) and number of layers of logic. The proposed technique practically obviates the need for more sophisticated analysis of channel phaseshifts as it takes advantage of simple logic simulation of LFSRs. Therefore, it can be employed in BIST automation tools to synthesize phase shifters. The conducted experiments also indicate that heuristic methods of phase shifter design used so far may produce unacceptable results, i.e., phase shifters with insufficient channel separations. These results may be unacceptable even if obvious shifted combinations of XOR taps are avoided.

As described herein, a phase shifter structure can be obtained in a very reasonable amount of CPU time while still allowing several trade-offs between the circuit complexity, the ability to drive a large number of scan chains, and the synthesis running time. Furthermore, its very natural integration with automated design tools with no constraints on the LFSR type, a characteristic polynomial used, and a requested channel separation, makes the proposed approach a very attractive solution for a wide variety of BIST applications.

Bibliography

1. P.H. Bardell, “Design considerations for parallel pseudorandom pattern generators,” *Journal of Electronic Testing: Theory and Applications*, vol. 1, No. 1, pp. 73-87, 1990.
2. P.H. Bardell, “Discrete logarithms: A parallel pseudorandom pattern generator analysis method,” *Journal of Electronic Testing: Theory and Applications*, vol. 3, No. 1, pp. 17-31, 1992.
3. P.H. Bardell and W.H. McAnney, “Pseudorandom arrays for built-in tests,” *IEEE Trans. Comput*., vol. C-35, No. 7, pp. 653-658, 1986.
4. P.H. Bardell, W.H. McAnney, and J. Savir, *Built-in Test for VLSI: Pseudorandom Techniques*, John Wiley & Sons, 1987.
5. W.W. Peterson and E.J. Weldon, Jr., *Error Correcting Codes*, Cambridge, MA: MIT Press, 1972.
6. J. Rajski and J. Vasudevamurthy, “The testability-preserving concurrent decomposition and factorization of
7. Boolean expressions,” *IEEE Trans. CAD of IC*, vol. CAD-11, No. 6, pp. 778-793, 1992.
8. J. Rajski and J. Tyszer, “Design of phase shifters for BIST applications,” *VLSI Test Symposium*, pp. 218-224, 1998.
9. J. Rajski, N. Tamarapalli, and J. Tyszer,