Testing of Uncustomized Segmented Channel Field Programmable Gate Arrays

Tong Liu
Dept. of Computer Science
Texas A&M University
College Station, TX 77843

Wei Kang Huang
Dept. of Electronic Engineering
Fudan University
Shanghai 200433, P.R. China

Fabrizio Lombardi
Dept. of Computer Science
Texas A&M University
College Station, TX 77843

Abstract

This paper presents a methodology for production-time testing of (uncustomized) segmented channel field programmable gate arrays (FPGAs) such as those manufactured by Actel [1]. The principles of this methodology are based on configuring the uncommitted modules (made of sequential and combinational logic circuits) of the FPGA as a set of disjoint one-dimensional arrays similar to iterative logic arrays (ILAs). These arrays can then be tested by establishing appropriate conditions such as constant testability (C-testability). A design approach is proposed. This approach is based on adding a small circuitry (consisting of two transistors) between each pair of uncustomized modules in a row for establishing the ILA configuration as a one-dimensional unilateral array. It also requires the addition of a further primary pin. Features such as number of test vectors and hardware requirements (measured by the number of additional transistors and primary input/output pins) are analyzed; it is shown that the proposed design approach requires a considerably smaller number of test vectors (a reduction of more than two orders of magnitude) and hardware overhead for the testing circuitry (a reduction of 13.6%) than the original FPGA configuration of [1]. The proposed approach requires 8+2n_f vectors for testing the uncommitted FPGA of [1], where n_f is the number of flip-flops (equal to the number of sequential modules for the FPGA of [1]) in a row of the FPGA.

Index terms: *testing, FPGA, constant testability, manufacturing.*

[0]This research supported by a grant from the Texas ATP.

1 Introduction.

Field Programmable Gate Arrays (FPGAs) are widely used for rapid prototyping and manufacturing digital systems using ASIC design [3, 4]. FPGAs are available commercially and two general types can be distinguished: SRAM-based and segmented channel FPGAs [1, 2]. Research in FPGAs has encompassed many aspects, such as technology mapping, routing and placement [4, 8].

Recently, interest in yield and fault tolerance by reconfiguration has been reported in the technical literature [5, 6, 7, 11, 12]. These approaches implement placement in the presence of faulty modules such that a FPGA can be still utilized. However, they rely on an effective diagnosis procedure such that the status of all modules can be assessed prior to reconfiguration.

Testing of FPGAs has not been analyzed in depth in the literature []16,19]. In [15], testing of segmented channel FPGA has been discussed. This approach relies on a modified scan procedure to test sequentially every module in a FPGA, thus guaranteeing 100% fault coverage under a single stuck-at fault assumption. This requires a large number of tests and does not fully exploit the modularity in the FPGA to further reduce testing at production-time.

The objective of this paper is to propose a new methodology for testing segmented channel FPGAs. This methodology is based on connecting the modules on each row of a FPGA as a one-dimensional unilateral array such that the FPGA can be tested as a set of disjoint one-dimensional arrays, each array tested by establishing appropriate conditions such as C-testability [13, 14]. This results in a considerable reduction in both number of test vectors and testing circuitry.

This paper is organized as follows. Section (2) introduces the preliminaries, inclusive of notation and assumptions. Section (3) describes the basic principles of the proposed approach. Sections (4) and (5) deal with testing of a combinational and sequential mod-

Figure 1: A Row in the FPGA of [1].

ule respectively. Section (6) introduces the design for constant testability in an array made of combinational modules. Section (7) and (8) extend this design to an array with sequential modules as well as to a FPGA. Conclusions inclusive of a comparison with the design of the FPGA of [1], are presented in the last section.

2 Preliminaries.

A field programmable gate array (FPGA) can be thought as an homogeneous two-dimensional array in which modules can be programmed to implement combinational as well as sequential logic functions [4, 9]. Modules are separated by a programmable interconnection network; the interconnection network may consists of either programmable connector and switching modules [2], or a series of horizontal/vertical routing tracks and segments with programmable devices (usually in an antifuse configuration) [8, 9]. Customization is achieved by connecting the desired modules and programming the augmented interconnection network.

In this paper the FPGA configuration of [1] is analyzed and in particular, testing of this type of FPGA prior to customization is considered, i.e. testing at production-time for manufacturing purposes. In this FPGA, every row consists of I/O modules (generally denoted as I-modules), programming modules, C-modules and S-modules (as shown in Figure 1). The C-module implements the following combinational function:

$$Y = \overline{S_1} \times \overline{S_0} \times D_{00} + \overline{S_1} \times S_0 \times D_{01} + S_1 \times \overline{S_0} \times D_{10} + S_1 \times S_0 \times D_{11} \quad (1)$$

where $S_0 = A_0 \times B_0$ and $S_1 = A_1 + B_1$. Note that + ($\times$) stands for the logical OR (AND) operator.

The C-module is shown in Figure 2 and consists of a multiplexer with four data inputs. Figure 3 gives the logic and block diagrams of multiplexers with two data inputs; Figure 4 shows the block diagram of multiplexers with four data inputs.

The S-module implements the same Y function (except that $S_0 = A_0$ only, because B_0 is used as the reset input) followed by the sequential block. The sequential block implements either a D-type flip-flop, or a transparent latch [1]. This block can be fully transparent such that the S-modules can be used to implement purely combinational functions as for the C-modules.

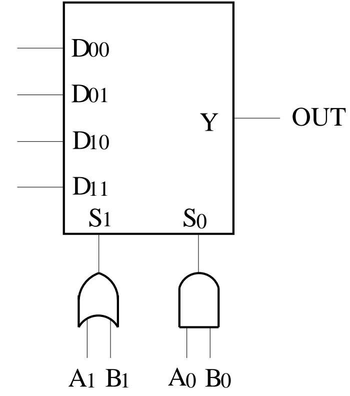

Figure 2: Block Diagram of a C-Module.

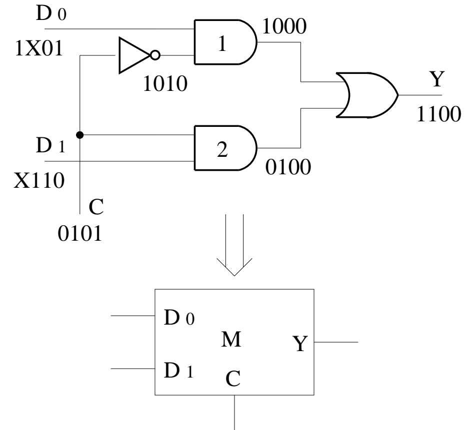

Figure 3: Logic and Block Diagrams of a Multiplexer with Two Data Inputs.

The following assumptions are made throughout this paper.

(a). The single stuck-at fault is assumed. This may occur in the modules (S- and C-modules) as well as any additional circuitry required for testing (note that faults in the interconnection network inclusive of bridging faults, can be fully diagnosed using relatively simple test sets due to its regularity [15]). Note that this assumption can be relaxed to multiple faults as a single stuck-at fault per row of the FPGA with no implication on the correctness of the proposed approach.

(b). The array is uncommitted, i.e. prior to customization and the programming process.

(c). The proposed approach tests only the logic modules (C- and S-modules) of the FPGA.

3 Basic Principles.

The proposed approach is based on the following two considerations:

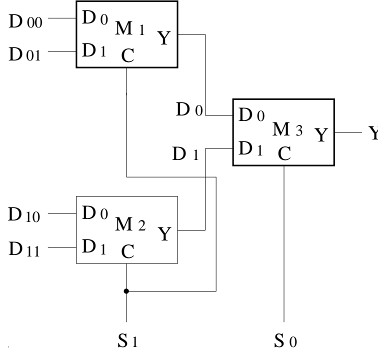

Figure 4: Block Diagram of a Multiplexer with Four Data Inputs.

1. If the Y output of a module (either C- or S-module) is connected to some inputs of the right (or left) neighbor module along the row, then the whole row of modules in the FPGA becomes a one-dimensional (unilateral) array. This array can be tested using established techniques as applicable to ILAs [14].

2. As each row of modules is the same in a FPGA, then they can be tested disjointly, i.e. the FPGA can be fully tested by testing all rows of modules at the same time.

The above considerations under a single stuck-at fault assumption are used to prove that a row of modules in the FPGA reconfigured as a one-dimensional array, is C-testable (using a constant number of tests irrespective of the number of modules, i.e. constant testability) [13] provided all S-modules are used to implement the same combinational function as the C-modules and rows are slightly modified to accommodate two transistors between each pair of modules. The total number of test vectors required for fully testing this type of array is 8. Also, it is proved that in general, the one-dimensional array (made of the modules in the FPGA) can be tested by pipelining the test vectors (for a pipeline made of S-modules). In this case, the total number of vectors to fully test the row is $(8+2n_s)$, where n_s is the number of S-modules in a row of the FPGA. This is the same number of vectors to test the whole FPGA.

4 Testing of the C-module.

Testing of a C-module can be analyzed by first considering the testing of a multiplexer. A multiplexer with two data inputs can be fully tested using only four vectors under a single stuck-at fault assumption [17]. These four vectors are given in Table (1), where

Table 1:

Test Vector	t_1	t_2	t_3	t_4
C	0	1	0	1
D_0	1	x	0	1
D_1	x	1	1	0
Y	1	1	0	0

Table 2:

Test Vector	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
S_1	0	0	1	1	0	0	1	1
D_{00}	0	1	1	x	1	x	x	x
D_{01}	1	x	0	1	x	x	x	x
D_{10}	1	x	x	x	0	1	1	x
D_{11}	x	x	x	x	1	x	0	1
S_0	0	0	0	0	1	1	1	1
D_0	0	1	0	1	1	x	x	x
D_1	1	x	x	x	0	1	0	1
Y	0	1	0	1	0	1	0	1

the test set $T=\{t_i\}$, i=1, ... , 4. When t_2, t_3 and t_4 are applied to the two inputs, the test sequence (11,01,10) is applied to the AND gate G_2. The upper input of the OR gate (the same as the output of the AND gate G_1) is (0,0,0). The effect of a stuck-at fault can be propagated through the OR gate. Therefore, the AND gate G_2 is fully tested. The AND gate G_1 is tested similarly when t_1, t_2 and t_3 are applied (the NOT gate is also tested at this time). The OR gate is tested when t_1, t_2 and t_3 (or t_1, t_2 and t_4) are applied and the test sequence (10,01,00) appears to its two inputs. Note that a stuck-at fault on the primary input C can be detected by t_3 and t_4, because only one sensitized path exists from C to Y.

Equivalently, it is well known that a multiplexer with four data inputs can be fully tested by eight test vectors under the single stuck-at fault assumption [17]. These vectors are given in Table (2). When t_1, t_2, t_3 and t_4 are applied, then in Figure 4 $(C, D_0, D_1) = (001, 01X, 110, 1X1)$, where X stands for don't care; these are required to fully test a multiplexer with two data inputs and are applied to the multiplexer 1 (denoted as M_1). At the same time, 0 is applied to S_0 (the C input of multiplexer 3, M_3) such that $Y = D_0$ (the output of M_1). Therefore, the effect of a stuck-at fault can be propagated to the Y output and M_1 is fully tested. When t_5, t_6, t_7 and t_8 are applied, M_2 is fully tested in a similar fashion. Note that faults on the primary input S_1 can be detected by either t_1 and t_3, or t_5 and t_7, because only one sensitized path from S_1 to the primary output Y exists. When t_1, t_2, t_5 and t_6 are applied, M_3 is fully tested because $(C, D_0, D_1) = (001, 01X, 110, 1X1)$ as required to fully test a multiplexer with two data inputs.

The above considerations can be used to prove that

Table 3:

Test Vector	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
A_1	0	0	0	1	0	0	1	0
B_1	0	0	1	0	0	0	0	1
D_{00}	0	1	1	x	1	x	x	x
D_{01}	1	x	0	1	x	x	x	x
D_{10}	1	x	x	x	0	1	1	x
D_{11}	x	x	1	x	1	x	0	1
A_0	0	1	1	0	1	1	1	1
B_0	1	0	0	1	1	1	1	1
D_1	0	1	0	1	0	1	0	1

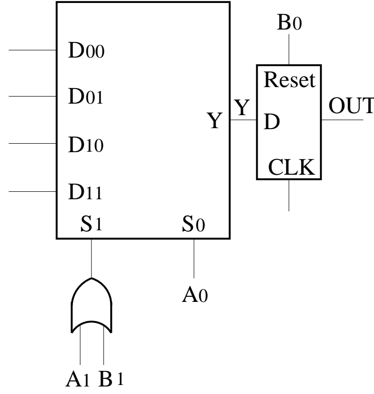

Figure 5: Block Diagram of a S-module.

to fully test a C-module under the single stuck-at fault assumption, only 8 vectors are required (as shown in Table (3)). Note that there is a sensitized path from S_1 to Y for t_1, t_3, t_5 or t_7 to test a multiplexer with four data inputs. In a C-module, $S_1 = A_1 + B_1$. Therefore, the only requirement for testing the OR gate in a C-module is to apply the test sequence $(00, 01, 10)$ to (A_1, B_1) for t_1, t_3, t_5 or t_7 to test the multiplexer. The AND gate in the C-module can be tested by applying the test sequence $(10, 01, 11)$ to (A_0, B_0) for t_1, t_3, t_5 or t_7 to test the multiplexer. However in t_3, D_{11} must be 1 instead of X such that $D_1 = 1$ in M_3. This is required for establishing a sensitized path from S_0 to Y in M_3.

5 Testing of the S-module.

As mentioned previously in Section (3), the S-module implements the same Y function as the C-module (except that $S_0 = A_0$ because B_0 is used for reset) and the Y output is connected to the sequential block as shown in Figure 5. The sequential block can implement either a D-type flip-flop or a transparent latch [1]. It can also be fully transparent such that the S-modules can be used to implement purely combinational functions as the C-modules.

The S-module (whose block diagram is given in Figure 5) can also be tested by eight vectors given in Table (4). Note that B_0 is not present in Table (4) and S_0=A_0 in Figure 5. Therefore, the eight vectors can fully test the C-module block in a S-module. The Y output of the C-module block takes both boolean values when the eight vectors are applied. Therefore, the sequential block (acting as either a D-type flip-flop, or latch) can be tested using the above vectors as long as a clock signal is applied to the module for every vector in T. To test the reset function, a reset signal must be applied to B_0 after the eighth vector (note that at this time, the output is 1). In the following discussion, for sake of simplicity the reset function is not explicitly considered.

Table 4:

Test Vector	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
A_1	0	0	0	1	0	0	1	0
B_1	0	0	1	0	0	0	0	1
D_{00}	0	1	1	x	1	x	x	x
D_{01}	1	x	0	1	x	x	x	x
D_{10}	1	x	x	x	0	1	1	x
D_{11}	x	x	1	x	1	x	0	1
A_0	0	0	0	0	1	1	1	1
OUT	0	1	0	1	0	1	0	1

6 C-Testable Design of a C-Module Array.

A C-module (one-dimensional) array can be constructed by connecting the Y output of each ith module to one of the inputs of the $(i+1)$th module ($i = 1, 2, ..., n_c - 1$, where n_c is the number of C-modules). However, this type of array is not C-testable because there is no single input which always has a sensitized path from the input to the primary output of the C-module when any of the eight vectors is applied to the C-module [15]. Hence, controllability and observability cannot be guaranteed within the array to satisfy the conditions for constant testability [14]. Therefore, an arrangement is proposed to allow the Y output of each ith module to be connected to either A_0 or B_0 of the next module, thus accomplishing C-testability of this array. Figure 6 shows such arrangement. The A_0 and B_0 inputs of a module can be connected to the output of the previous module by using two transistors and an appropriate signal to control them. When one input of a module is connected to the output of the previous module, then all other inputs of this module are considered as inputs in the vertical direction (through the vertical routing tracks of the FPGA [10]). Note that the additional transistors are ON during testing only.

The vectors given in Table (3) must be modified to

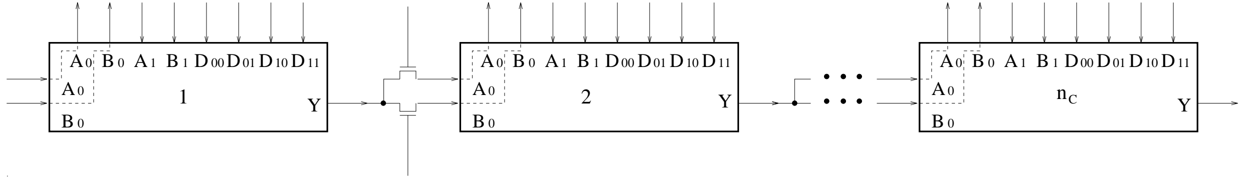

Figure 6: Construction of the C-Module Array.

Table 5:

Test Vector	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
A_1	0	0	0	1	0	0	1	0
B_1	0	0	1	0	0	0	0	1
S_1	0	0	1	1	0	0	1	1
D_{00}	0	1	1	0	1	x	x	x
D_{01}	1	0	0	1	x	x	x	x
D_{10}	1	x	x	x	0	1	1	x
D_{11}	x	x	1	x	1	x	0	1
D_0	0	1	0	1	1	x	x	x
D_1	1	0	x	x	0	1	0	1
A_0	0	1	0	1	1	1	1	1
B_0	1	0	1	0	1	1	1	1
S_0	0	0	0	0	1	1	1	1
Y	0	1	0	1	0	1	0	1

test an array made of C-modules. In t_2, the value of D_{00} is changed from X to 0 so that there exits a sensitized path from S_1 to the primary output Y. In t_3, the values of (A_0, B_0) are changed from (1,0) to (0,1) such that Y=0 when A_0=0. In t_4, the values of (A_0, B_0) are changed from (0,1) to (1,0). such that Y=1 when B_0=0. The new test vectors are given in Table (5). For convenience, some of the signal values inside a module are also presented. When the A_0 of a module is connected to the Y output of the next module, then t_1, t_3, t_6 and t_8 must be applied. As A_0 and Y of the same module are equal (that is, the Y output of the module can regenerate the input value required for testing its neighbor module), then the same test vector can be applied to all modules in the array, thus satisfying the controllability requirement for C-testability [13]. When the B_0 of a module is connected to the Y output of the next module, then t_2, t_4, t_5 and t_7 are applied. As B_0=0 and Y=1 (when t_2 or t_4 is applied) and B_0=1 and Y=0 (when t_5 or t_7 is applied), therefore t_2 and t_4 (t_5 and t_7) are applied to the even numbered modules, while t_5 and t_7 (t_2 and t_4) are applied to the odd numbered modules (the Y output of the module can also regenerate the correct input value as required for testing its neighbor module). Observability is also satisfied. Therefore, all modules can be tested simultaneously using a constant number of tests irrespective of their number, thus the array is C-testable [13, 14].

7 Testable Array Design.

In Section (6), testing of an array made of only C-modules, has been described. It is obvious that if all S-modules in a row of the FPGA implement the same combinational functions, then the row can be tested as a C-module array. A further feature for the array with S-modules is that $S_0 = A_0$, i.e. there is no AND gate and A_0 is directly connected to S_0. Note that during the testing process of the C-module array, the S_0 of a module is connected to the A_0 of the next module for t_1, t_3, t_6 and t_8 and is connected to B_0 of the next module for t_2, t_4, t_5 and t_7. Therefore, for testing a row in the FPGA with C-modules and S-modules, the array can be constructed by connecting A_0 (S_0) of every S-module to the Y output of its neighbor module (S-module or C-module) through a transistor, while the A_0 and B_0 of the C-modules are connected in the same way as in the construction of the C-module array of Section (6).

For testing the one-dimensional array consisting of a row of C-modules and S-modules in the FPGA (provided the S-modules implement combinational functions), the test set is the same as for testing the one-dimensional array made of a row of C-modules. However, the transistors which connect the S_0 of the S-modules to the Y output, must be ON during the test process. The only difference in the testing procedure is to account for the sequential function of the S-modules. If the D-type flip-flops (or transparent latches) of the S-modules are enabled, then the row of the FPGA can be tested using a pipeline technique. Assume that there are n_f flip-flops in a row of the FPGA. Let the C-modules and combinational blocks in the left (right) of ith flip-flop (or latch) be referred to as the ith ((i+1)th) combinational *cluster.* Testing of this type of one-dimensional array is accomplished in phases.

Testing of Combinational Modules and Blocks.

a In the first phase, the A_0 of every C-module is connected to the Y output of its neighbor module and t_1, t_3, t_6 and t_8 are applied.

b In the second phase, the B_0 of every C-module is connected to the Y of its neighbor module and t_2, t_4, t_5 and t_7 are applied.

Testing of Sequential Modules.

c The S-modules are tested by pipelining the tests.

For simplicity, the vectors in each of the above phases are reordered and generically denoted as as v_1, v_2, v_3 and v_4. When pipelining is employed for testing the one-dimensional array, v_j is applied to the ith cluster, while v_{j-1} is applied to the $(i+1)$th cluster at time t and v_{j+1} is applied to the ith cluster (in the a same fashion, v_j is applied to the $(i+1)$th cluster) at time $(t+1)$.

Let the number of test vectors in Phase 1 (2) be denoted as $n_{1v}(n_{2v})$; then, the test time for applying the $n_{1v}(n_{2v})$ vectors to every cluster in a row for Phase 1 (2) is $(n_{1v}+n_f)+((n_{2v}+n_f)$ clock periods. Therefore, the total time required to test a row in the FPGA is $(n_{1v}+n_{2v}+2\times n_f)$, or (n_v+2n_f) clock periods, where $n_v=(n_{1v}+n_{2v})$ is the total number of vectors for testing an array.

8 FPGA Testing.

In the previous section, it was shown that a row (made of C- and S-modules) in the FPGA can be tested as a one-dimensional array in (n_v+2n_f) clock periods. As all rows in the FPGA can be tested in parallel, then they can also be tested in (n_v+2n_f) clock periods. Note that the additional circuitry between modules is disabled once testing is completed and the FPGA can be customized; also this circuitry does not affect the performance features of the original FPGA of [1].

In the original FPGA design [1], the testing procedure consists of testing the FPGA row by row using a modified scan [15]. The vectors are applied to each module (C-module or S-module) in parallel. The output of each module is loaded through a shift register placed at the top of the FPGA (as in the original FPGA design of [1]), then the outcomes of the tests are shifted out to a primary output pin, as shown in Figure 7. The total test time required for testing a row in a FPGA, is $(n_v\times(n_c+n_s))$ or $n_m\times n_v$ clock periods, where n_c and n_s are the number of C-modules and S modules respectively, and $n_m=(n_c+n_s)$, i.e. the total number of C-modules and S-modules in a row of the FPGA. Assume there are n_r rows in the FPGA. Note that as n_m is approximately equal to $2n_f$, then the total time for testing the FPGA of [1, 15] is given by $n_m\times n_v\times n_r$ clock periods.

It is then possible to compare the time required for testing the proposed testable FPGA with the time required to test the original FPGA of [1]; it is easy to prove that the proposed testable design can save test time by a factor of $(n_m\times n_v\times n_r)/(n_v+2n_f)$. Note

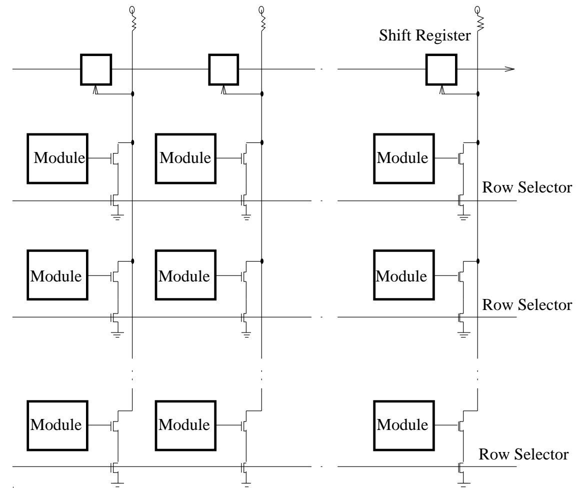

Figure 7: Test Scheme in the Original FPGA.

that C-testability in the presence of a fault will guarantee that the output of the last module of the row will be different from the correct value, thus accomplishing diagnosis of the FPGA.

Consider as an example, the FPGA chip A1280A of Actel 2 [1]; this FPGA has 18 rows and 82 columns. Note that four columns are used for I-modules and a column is for B-modules (these modules are not logic modules and are not tested by the proposed approach). Therefore, $n_r=18$, $n_m=77$. As $n_f=n_s=n_m/2$, then $(n_m\times n_v\times n_r)/(n_v+2n_f)=(77\times 8\times 18)/(8+77)=130$, i.e. a saving by a factor of almost 130 times.

Consider now the average hardware overhead in a module as modified for testability purposes in the proposed approach. In our testable design, two transistors are required for each C-module and a transistor for each S-module. The total number of transistors required in the proposed approach is $(2n_c+n_s)\times n_r$ or $(n_c+n_m)\times n_r$. Three pins are needed to control these transistors (if a OR gate is used, the number of pins is reduced to two). Assume that a NOT gate consists of two transistors, an NAND and NOR gate consist of 4 transistors respectively, a multiplexer and XOR gate consist of 12 transistors respectively. A D-type flip-flop consists of 22 transistors. Therefore, there are 44 transistors in a C-module and 62 transistors in a S-module approximately. The hardware overhead for the proposed testable design per module is almost 3%.

Note that in the original design, each C-module (or S-module) requires two transistors to connect its output to the shift register (as shown in Figure 7). Hence, the total number of transistors required for testing the FPGA is $2\times n_m\times n_r$. The hardware overhead per module is 4% approximately. Furthermore, there is an additional shift register for storing and shifting out the

Table 6:

	A1280A	Proposed Design
# of Transistors	87624	75690
# of Test Vectors	11088	85
# of Added Pins	2	3

test outcomes using a modified scan procedure [15]. A control circuitry is also required for row selection; this can be another register with n_r flip-flops or a decoder and a register with $\log(n_r)$ flip-flops. The shift register on the top of the FPGA consists of $(n_c + n_s)$ or n_m flip-flops. A flip-flop used to construct registers usually consists of eight transistors. Then the number of transistors required for this register is $8 \times n_m$. A primary pin is required for the output of the shift register. If another register with n_r flip-flops is required, then the total number of transistors is $8 \times n_r$ (a primary input pin is also required for this register). The total number of transistor required for testing the FPGA is $(8 \times n_r + 8 \times n_m + 2 \times n_m \times n_r)$. Hence, the proposed testable design requires also a considerably smaller number of transistors than the original design of [1].

9 Conclusions.

In this paper, a testable design method for uncustomized segmented channel FPGAs has been proposed. In Table (6), a detailed comparison of hardware overhead and test time is given using the A1280A of [1] as an example. As the exact data on the internal detailed structure and layout of the A1280A are not available outside the manufacturer, the results given in Table (6) are only a good approximation based on a transistor count and the number of required clock cycles.

However, it is clear that the proposed design method can significantly reduce the testing time, while requiring less hardware too (albeit one more primary pin is needed). In the analysis, the total number of transistors includes only the transistors in C-modules, S-modules and the circuits to test them. It does not include the transistors in I-modules, B-modules and other circuits such as probe related circuits and clock circuits. Note that these modules and circuits must be present in the proposed testable design as in the original FPGA configuration of [1].

References

[1] Actel Corporation, FPGA Data Book and Design Guide, Sunnyvale, 1994.

[2] Xilinx Inc., Programmable Gate Array Data Book, San Jose, 1991.

[3] Chan, P.K. and M.D.F. Schlag, "Architectural Tradeoffs in FPGA-based Computing Systems," *Proc. IEEE Workshop for Custom Comp. Machines,* pp. 152-161, 1993.

[4] Brown S., R.J. Francis, J. Rose and Z.G. Vranesic, "Field Programmable Gate Arrays," Kluwer Academic Publishers, Boston, Mass. 1992.

[5] Hatpori, F., et all, "Introducing Redundancy in FPGAs," *Proc. IEEE CICC,* pp. 7.1, 1993.

[6] Kelly, J. L. and P. A. Ivey, "A Novel Approach to Defect Tolerant Design for SRAM Based FPGAs," *Proc. ACM 2nd Int. Work. on FPGAs,* Berkeley, 1994.

[7] Narasimhan, J., K. Nakajima, C.S. Rim and A.T. Dahbura, "Yield Enhancement of Programmable ASIC Arrays by Reconfiguration of Circuit Placements," *IEEE Trans on CAD of ICAS,* Vol. CAD13, No. 8, pp. 976-986, 1994.

[8] Green, J., V. Roychowdury, K. Kaptanoglu and A. El Gamal, "Segmented Channel Routing," *Proc 27th IEEE/ACM DAC,* pp. 567-572, 1990.

[9] El Gamal, A., "Field Programmable Integrated Circuits, Overview and Future Trends," *Proc IEEE Int. Conf. on Comp. Des.,* pp. 2, 1992.

[10] El Gamal, A. et all., "An Architecture for Electrically Configurable Gate Arrays," *IEEE Journal of Solid State Circuits,* Vol SSC24, No. 2, pp. 394-398, 1989.

[11] Fawcett, B.K., "Taking Advantage of Reconfigurable Logic," *Proc. ACM 2nd Int. Work. on FPGAs,* Berkeley, 1994.

[12] Durand, S. and C. Piguet, "FPGA with Self-repair Capabilities," *Proc ACM 2nd Int. Work. on FPGAs,* Berkeley, 1994.

[13] Lombardi, F. and W-K Huang, "On an Improved Design Approach for C-testable Orthogonal Iterative Arrays," *IEEE Trans. on CAD of ICAS,* Vol. CAD7, No. 5, pp. 609-615, 1988.

[14] Friedman, A.D., "Easily Testable Iterative Arrays," *IEEE Trans on Comput,* Vol. C22, NO. 12, pp. 1061-1064, 1973.

[15] Al-Ayat K, R. Chan, C.L. Chan and T Speers, "Array Architecture for ATPG with 100% Fault Coverage," *Proc. IEEE Int. Work. on DFT in VLSI Systems,* Hidden Valley, pp. 213-226, 1991.

[16] Ghewala, T., "CrossCheck: a Cell Based VLSI Testability Solution," *Proc. IEEE/ACM CAD,* pp. 706, 1989.

[17] Makar, S.R. and E.J. Mc Cluskey, "On the Testing of Multiplexers," *Proc. IEEE Int. Test Conf.,* pp. 669-679, 1988.