TEST METHODS OF PAL-, PLA-, AND MAPL-STRUCTURES

Karel VLČEK
Dept. of Electronics
FEI, VŠB-TU Ostrava
708 33 Ostrava-Poruba
Czech Republic
E-mail: karel.vlcek@VSB.CZ

Abstract

The architecture of various programmable logic arrays such as PAL (Programmable Array Logic), PLA (Programmable Logic Array) and MAPL (Multiple Array Programmable Logic) differ slightly in interconnection. The introduced types of devices are called PLD (Programmable Logic Devices). It is a bulk of programmable AND functions (product terms), and OR functions. The whole circuit structure is completed by input/output or dedicated output macrocells allowing to do the minimization of product term number. PLD's internal AND-array, and OR-array differs from the discrete logic AND, and OR devices whereas the functions are similar. The troubleshooting of these devices differs too.

1. Introduction

Familiarly known fault models *t0, t1*, and *z* are not sufficient to describe various electrical defects of devices. For that reason further models in troubleshooting are applied. There are four classes of models of behavioral describing: *G* (Growth) if the number of implicants is growing, and *S* (Shrink) if it is shrinking [1]. The *G* and *S* are faults of product terms. *D* (Disappearance) and *A* (Appearance) are faults of programmable OR-array functions.

2. The Various Architectures

As it was pointed: PAL-, PLA- and MAPL-architecture differs slightly only. The basic differences and its influence on the testing method will be introduced.

2.1 PAL-Architecture

The PAL-architecture is characterized by a programmable AND-array and a fixed number of independent product terms per output. This architecture is the most popular one today. Its high speed, high fan-in, and moderate product term number per output make the PAL-architecture most widely used in PLDs.

The high speed is the reason for the application; the limitation is its relatively low density. The both properties influence one another. It is impossible to scale up the size of the PAL-architecture without sacrificing performance and power. Unfortunately some applications (except decoders and a simple random logic) need more product terms than the PAL-structure satisfies. The architectures which are related to the characteristics of the application complicate the design. More information and more details of architecture of input/output macrocell properties are necessary in reducing system parts. It is easy to choose the model of troubleshooting models *t0, t1* and *z* and the models *G* and *S* as a special models of product term faults.

2.2 PLA-Architecture

The PLA-, FPLA-structures (Field Programmable Logic Array) differ from the PAL-one: both the AND-array and the OR-array are programmable. This means that all product terms are available to all outputs: several outputs can share the same product terms. The PLA-architecture is an ideal programmable logic array structure. The utilization of product terms in PLA-structure is generally better than in PAL-one.

The disadvantage of PLA-structure: it is much slower. Signals must go through two programmable arrays from inputs to outputs. Time delays are influenced by programmable point parameters. Designers have tended to avoid PLA-structures in speed-sensitive applications. However, the biggest advantage of the PLA-architectures is its higher density. The PLA-structures are ideally suited for state machines. The PLA-architecture provides ample product terms to the user. This allows complex state machines to be implemented in a single device.

Some characteristic properties of PAL- and PLA-structures:

- the possibility to be designed at the state machine level;
- the design benefits are the maintenance and automatic documentation of the design process;
- all outputs have identical timing.

The second property is satisfied by the relations between the basic disjunctive normal form (NF-AND/OR) and Peirce's normal form (NF-NOR/NOR) used with CMOS technology. From three another forms can be derived the disjunctive normal form (DNF). Let it will be:

$$A + B = /A \;?\; /B \qquad (1)$$

$$/(A \,.\, B) = A \;?\; B \qquad (2)$$

The translations of these forms are in the diagram [2]:

$$\begin{tikzcd}
\text{DNF} \arrow[r] \arrow[d] & \text{NF ?/?} \arrow[d] \\
\text{NF ?/+} & \text{NF +/?}
\end{tikzcd}$$

The last property is an advantage for design and for diagnostics too. The identical timing derived from the single source of synchronization is called the Huffman's model architecture, and it allowed the designer to create the test vector by the regular method of troubleshooting. The diagnostics of PLA-structures must consider all the fault models of testing: *t0, t1, z, G, S, D* and *A*.

The *input/output macrocell testing* pays attention to its special architecture: each device is completely tested by the manufacturer using numerous test patterns prior to shipping. Every programmable cell and every logic path through the device is fully tested because of EECMOS technology.

2.3 MAPL-Architecture

The MAPL devices are new higher density PLDs. The MAPL integrates multiple FPLAs (Field Programmable Logic Arrays), allowing for easy implementation of complex state machines, controllers, microinstruction sequencers, bus interfaces and *general synchronous logic* designs. The MAPL device architecture is functionally equivalent to a large continuous FPLA, having a higher number of product terms. All the product terms belong to some individual FPLA planes or pages, consisting of both a programmable AND-array as well as a programmable OR-array with the same configuration. Devices can be shared among all outputs and macrocells. There is no restriction on the number of product terms that can be connected to an output, no limiting switch matrix and no time consuming expander arrays to deal with.

In addition to the first generation of MAPL devices, the second generation MAPL *incorporates a separate PAL block with combinational I/O to handle asynchronous and decode functions.* The second generation MAPL devices allow for easy and cost effective integration of complex subsystems consisting of synchronous and asynchronous elements. There are individual pages in FPLA array, giving a total of product terms, feeding I/O logic macrocells (IOLMCs). The IOLMCs are individually configurable to provide register or combinational outputs, with programmable polarity. Alternatively they may provide a buried feedback path and the I/O pins may be used as dedicated inputs. There is a choice of clocks, output enable and reset conditions. Flexible macrocell architecture implements both DE-type and JK-registers with clock enables. This alleviates the need for software transformations or additional product terms to implement hold and toggle functions.

The architectures based on PAL block can be shared among all outputs and macrocells since the OR-array is user programmable. Therefore duplications of product terms for each output or transition are not required. Any input can be routed to any output within the array. The internal PAL array and control array are always active and available. The FPLA array of MAPL device is partitioned into pages. The only current page is active at any time, the remaining pages being effectively deactivated.

3. Methods of Testing

The classification of testing may be very difficult, when we consider all methods for LSI circuits, each with its own way of generating test patterns, and evaluating the responses obtained from Device under Test (DUT). The testing can be divided into two categories: Concurrent Testing and Explicit Testing [3].

3.1 Concurrent Testing

Concurrent testing is to be carried out under continuous operation. Concurrent testing approaches provide the following advantages:

- The fault is detected in real time. The user can recognize the correctness of output results when the fault coverage is provided by using the error detecting code.
- Transient faults may be detected. Faults may occur during normal operation; they are detected if they cause any faulty data pattern.
- The test equipment, and the test pattern generation are eliminated during the life of the system, since the data patterns used in the normal operation serve as test patterns.

On the other hand, the concurrent testing approach have some problems that limit its usage in PLD testing:

- The design of circuit for concurrent testing is much more complicated task than designing a similar circuit that will be tested explicitly.
- The implementation of parity checkers require additional hardware circuitry as well as additional storing elements. Transferring data ways are required too.
- Defects may be located in places that are not exercised during the normal function. The faults of these defects will not be detected. The most error detection codes have a limited capability for detecting multiple faults.
- Additional input/output pins are required to detect the error in information signals in a PLD chip.
- No control of over-voltage or timing parameters is provided. The DUT cannot be tested under marginal timing and electrical conditions.
- The fault coverage of concurrent testing is less than that provided by explicit testing methods.

3.2 Explicit Testing

The explicit testing methods are characterized by the separating of the testing process from normal operation. There are three steps involved in the explicit testing process:

- The first step can be called *generating of the test patterns*. The goal of this step is to produce the input patterns of DUT to test it under different modes of operation while trying to detect any existing fault.
- The test patterns generated in the first step are applied during the second step. There are two ways to accomplish this step. The first one is *external testing* - the use of special test equipment to apply the test patterns externally. The second one is internal testing: the test patterns are forced into the DUT to execute a *self testing procedure*. External testing gives better control over the test process.
- The final step is evaluating the responses from the DUT. The two goals are considered. The first is to detect errors, which indicate the existence of one or more faults. This process is called *"go/no-go test"*. The other is to localize the fault if one exists, in some replaceable module. This goal is the *"fault location testing"*.

Some PLDs have many thousands of gates. The gate level approach to the test generation is not very feasible. A new approach to functional level is needed. At the functional level the circuitry is considered as a black box with a well-defined function. If no fault model is assumed, then the tests derived must be either exhaustive or a rather *ad hoc* check of the functionality of the system. Exhaustive tests are impossible for even small systems because of the enormous number of possible states.

4. PLDs Test Pattern Generation

Four methods may be used to generate test patterns for PLDs:

- The *manual approach* was widely used for medium- and small-scale digital circuits. The formulation of the D-algorithm and similar algorithms eliminated the need for analyzing each circuit manually. The manual test generation techniques are not efficient for PLDs, especially for microcontrollers. Realizing that test generation has to be done economically, test designers are now moving in the direction of automatic test generation.
- *Algorithmic test techniques* are much more economical than manual techniques. The best designers can improve the fault coverage of the test. The problem of generating meaningful sets of tests directly from the functional description of the DUT has become increasingly important. Various test generation algorithms have been developed to detect different types of faults in memories. The PLD

architectures are regular as well as those of memories. The fault models describe the faulty behavior of the DUT without its implementation of *t0, t1* or *z* models. This model bypasses the gate and the flip-flop levels and directly describes the circuit blocks according to functions. Each module in PLD is modeled as a black box having a number of functions defined by a set of binary decision diagrams. An important feature of these diagrams is that they state exactly how the module will be simulated. These experiments describe the behavior of the module in one of its models of operation. It is suitable for use in automatic test generation. For the operations we give an algorithm which takes the set of experiments and the current state as parameters.

- *Logic simulation techniques* have been used in the evaluation and verification of new digital circuits. The simulator can simulate the behavior of the circuit under normal conditions as well as when any fault exists. The *fault simulation* interprets the behavior of a circuit under a fault or faults.
- The *random number generator* is used to apply random input patterns both to the DUT and to a copy of it known to be fault-free simultaneously. The results obtained from the two units are compared, and if they do not match, a fault in the DUT is detected. The restriction of this method is the dependency on the fault-free unit. There is no accurate measure of how effective the test is [4]. The amount of fault coverage is unpredictable. This method can be considered the simplest method for testing the DUT. Many variations of the random number generator were referred [5].

5. Microcell Testing

The specific problem of test application is the IOLMCs (Input/Output Logic Microcells) testing. The *register preload* feature allows IOLMC registers to be directly loaded with any desired data pattern. It also allows the present state of IOLMC to be examined regardless of tri-state control conditions. This simplifies the testing of the device after programming. This allows a complete verification of sequential logic circuits, including states that are normally impossible. Register preload is not an operational mode and is not intended for board level testing because elevated voltage levels must be applied to the device. The programming equipment normally provides the register preload capability as part of its functional test facility.

The register preload algorithm is described for the test equipment other than approved PLD programming equipment. To preload the IOLMC registers, a sequence of data bits is shifted into the device on the SDIN (Shift Data INput), one bit for each IOLMC in which registered output has been selected. Non-registered IOLMCs are bypassed. The shift sequence is clocked by the rising edge of the DCLK input.

The data stream is shifted through the IOLMC with the lowest corresponding pin number. All remaining registered IOLMCs are loaded in ascending order. Therefore the first data bit in the series is ultimately loaded into the registered IOLMC with the highest corresponding pin number. As the data series is shifted into the SDIN, the contents of all registers are shifted "upward" and out onto the SDOUT (Serial Data OUTput). All of the used registers in the PLD can be preloaded, including the input, I/O, and state registers.

6. Boundary-Scan Test

In respect of the IEEE Std 1149.1 the recommendations of the standard are not fully accepted. The diagnostic hardware can be used as Boundary-Scan Test one when only one device is tested. It is not intended for board level testing. If we consider the board level testing for PAL, PLA, and MAPL, we must complete the tested circuitry by further recommendations [6]. Some new types of the PLD such as FPGA (Field Programmable Gate Array) are fully compatible with the standard. This support circuitry can be concatenated to one or more chains to be tested in standard protocol on board or inter board level.

References

[1] Drábek, V.: Spolehlivost a diagnostika. VUT Brno, SNTL Praha (1986).

[2] Frištacký, N.: Logické systémy. ALFA Bratislava (1986).

[3] Hassan, S. Z., McCluskey, E. J.: Testing PLAs using multiple parallel signature analyzers. In: Proc. 13th Int. Symp. on Fault Tolerant Computing, Milano (1983), p. 422-425.

[4] Golan, P., Novák, O., Hlavička, J.: Pseudoexhaustive Test Pattern Generator with Enhanced Fault Coverage. IEEE Trans. Comp., vol. C-37, no. 4, April 1988, p. 496-500.

[5] Novák, O.: Pseudoexhaustive Test Sets Generated in LSFRs. Proc. IEE-ETC91 Conference, Munchen, Germany, (1991), p. 485.

[6] Bleeker, H., van den Ejinden, P., de Jong, F.: Boundary-Scan Test (A Practical Approach). Kluwer Academic Publishers. The Netherlands (1993).