The Identification of Registers in RTL Structures for the Test Application

Abstract

A highly efficient test schedule can be created for a given digital circuit if a proper test-scheduling algorithm is selected and if the circuit fulfils several criteria that affect the quality of a resulting test schedule significantly and independently on any scheduling algorithm. If a design for testability techniques are applied to the circuit structure, the way in which they are applied have a big impact on those circuit properties. Thus, it is feasible to deal with the relation between application of selected design for testability techniques and the quality of a resulting test schedule in detail. This is a wide research area. Our paper deals with the register selection technique through which a test will be applied. The paper presents a methodology for selecting registers for the test application in such a way that the cardinality of a set of selected registers is minimized and test resources allocated to functional units are shared in a maximum way. Proposed methodology is mathematically described, definitions are clearly illustrated and experimental results together with the future research perspectives are discussed.

1 Introduction

Today it is typical that modern complex designs are described at higher levels of the abstraction (e.g. register-transfer level - RTL). Although these designs help to reduce design cycle time, they pose several difficult test challenges. In most of cases, the test constraints are related to minimum test resources, test data traffic, test application time and power consumption. The goal is to prepare the test when required constraints are satisfied. The best way how to do it is to use a test-scheduling methodology.

To achieve short test time, it becomes important to study potential parallelisms in the application of test patterns to unit under test (UUT). As a UUT we can see different structures, SOC (System On Chip) or RTL componets can serve as an example. It can be stated generally that if more elements in a UUT can be tested in parallel then shorter test time can be gained. On the contrary, test resources sharing will result in longer test application time.

The methodology presented in this paper selects registers in the way allowing their sharing. It can be used for RTL structures consisting of functional units (*fui*), registers (*reg*) and multiplexers (*mux*).

The paper is organised as follows. First, the related works from the area of the test scheduling and selecting registers are described. Then, in Section 3, the proposed methodology is explained and the possibilities how the Hasse diagram could be used are described. Section 4 describes experimental results. The possible future work and conclusions are mentioned in Section 5 and 6.

2 Related Work

2.1 Test Scheduling

In [6], a test scheduling problem is identified as an open-shop scheduling problem, which is known to be NP-complete and the use of heuristics are therefore justified. Recently, numerous methodologies for the test scheduling were developed, many of them for SOCs. In [15], an integrated framework for the design of SOC test solutions is presented. It deals with test scheduling, TAM design, test sets selection and test resource placement, together with minimization of test application time and TAM considering constraints posed on test and power consumption. In [7], a test resource partitioning (TRP) technique that simultaneously reduces test data volume, test application time, and scan power is presented. Another method for solving the resource allocation and test scheduling problems in order of achieving concurrent test of core-based SOC designs is described in [10]. The main objective is to reduce test application time under the constraints of SOC pins and peak power consumption. A TAM named CAS-BUS that solves some of the new problems the test industry has to deal with is presented in [2]. CAS-BUS is compatible with IEEE P1500 standard proposal, is controlled by Boundary Scan features. The upcoming IEEE P1500 (SECT) standard proposes DfT solutions to alleviate it. The solutions based on this approach can be found in [16], [17].

2.2 Identification of Registers for Test Application in RTL Structures

Several approaches exist that suggest a modification of original circuit structure in order to obtain a circuit with the same functionality, but with improved selected properties, e.g. better testability, lower power-consumption, lower test-data volume etc. In the following text, we will concentrate on a brief presentation of today's selected approaches that try to modify original circuit structure in order to obtain a highly efficient test schedule for the circuit.

It is evident that if more functional units (*fui*) can be tested in parallel then shorter test time can be gained. Several approaches are used to model this fact – e.g., in [l1] a resource graph is used to model the system where an edge between a test and a resource indicate that the resource is required for the test. From the resource graph, a test compatibility graph (TCG) is generated, where each test is a node and an edge between two nodes indicates that the tests can be scheduled concurrently, i.e. that tests are compatible. Using TCG, the test-scheduling problem is reduced to

1. finding all the cliques of the TCG, and
2. solving the covering problem.

Another problem in the test scheduling is to find minimum test resources, because not all RTL circuit elements can be used for the test of each *fui*. The identification of minimum test resources leads to maximum sharing. The paper [11] presents a scheduling method for reducing the number of scan registers for acyclic structures. To estimate a number of scan registers during scheduling, provisional binding of operational units is proposed and a force-directed scheduling algorithm with the provisional binding is presented. In [14], a technique integrating test scheduling, scan chain partitioning and test access mechanism (TAM) design minimizing the test time and the TAM routing cost while considering test conflicts and power constraints is presented. Main features of the technique are (1) the flexibility in modelling the systems test behaviour and (2) the support for interconnection test of unwrapped cores and user-defined logic. [8] describes a technique for reordering of scan cells to minimize power dissipation that is also capable of reducing the area overhead of the circuit compared to a random ordering of the scan cells. For a given test set, proposed greedy algorithm finds the (locally) optimal scan cell ordering for a given value of a trade-off parameter(s). During our research activities, we have developed a methodology for selecting registers into partial scan in order to achieve a solution with a feasible trade-off among area overhead, pin overhead and testability constraints - our latest research results in this area are presented in [18], [12].

It can be concluded that in RTL designs registers play the major role during test application. In this paper, a new methodology for selecting registers for the test application is presented. The formal model - Hasse diagram and its theory are used. It allows to identify minimum number of circuit registers, through which test can be possibly applied.

3 The Proposed Methodology

3.1 Introduction

A new methodology of selecting registers for test schedulling in RTL circuit will be described in this chapter. Test schedule is developed for functional units (*fui*) and it determines which test resources will be utilised to test every *fui*. It is assumed that along i paths (*In [1] the i path concept was introduced in the following way: a structure S with an input port X and output port Y is said to have an identity mode (i mode of operation), if S has a mode of operation in which the data on port X is transferred (possibly after clocking) to port Y. Similarly, there is an identity transfer path (i path) from output port X of structure S1 to input port Z of structure S2, if the data at port X can be transferred unchanged to port Z*) existing in the circuit, test vectors can be transported from primary inputs (or from scan register if there is no connection to primary inputs) to the inputs of particular *fui*. Similarly, test responses can be transported from the outputs of *fui* to primary outputs (or output scan register). I paths existing in the circuit can be utilised for these transports. It is necessary to stress that the number of i paths leading to/from *fui* can be different for each *fui*. In a circuit under analysis, several i paths for particular *fui* can be identified. When a test schedule for the circuit under analysis is being developed, then the appropriate i path must be selected for the transport of diagnostic data, which can be provided by means of different criteria. The methodology presented in this paper takes into account the criteria of minimal number of registers which must be included into test resource set through which the test patterns will be applied and responses to them collected and observed. The number of selected registers can affect cost/testability trade-off significantly, as well as dynamic parameters can become worse. So it becomes to be reasonable to develop methodologies whose objective is to identify minimum number of registers through which the test will be applied. The drawback of such methodologies can be seen in the fact that the selected registers must be shared during the application of test to different *fuis*, thus increasing the test application time (the number of test phases increases as well [12]). Then, priorities must be defined – either minimum number of test phases or minimum number of registers used during test. Another possibility can be found in utilising optimising procedures based on reflecting both of these aspects.

In our research we concentrated on developing a methodology for the identification of a minimum number of registers (and primary connectors) through which the test will be applied. For this purpose we used the theory of Hasse diagrams. The algorithms, which are presented in this paper, can be utilised to solve the problem of the identification of registers for scan. The elements of an RTL circuit in Fig. 2 are subdivided into sets according to the function covered by the element. The circuit consists of four *fui* elements, set *FUI* = {fui_1, fui_2, fui_3, fui_4} to which test patterns will be applied. Primary inputs and outputs belong to the sets *PI* = {pi_1, pi_2, pi_3} and *PO* = {po_1, po_2}. Also, there is a set of circuit registers *R = {r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12, r13, r14, r15, r16}*, multiplexers MUX = {*mux1, mux2, mux3, mux4, mux5*}.

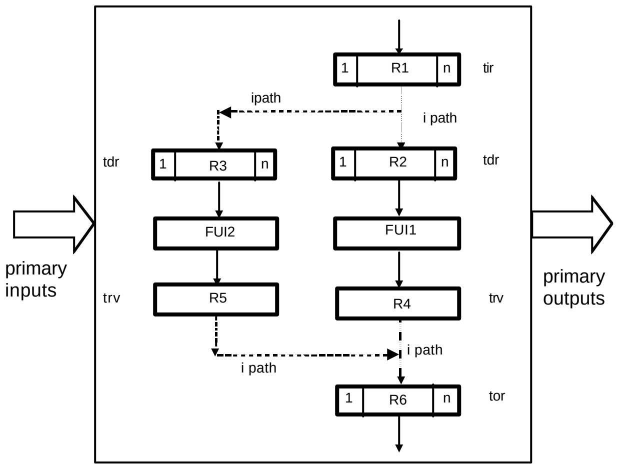

Fig. 1: Illustration of register roles in a circuit

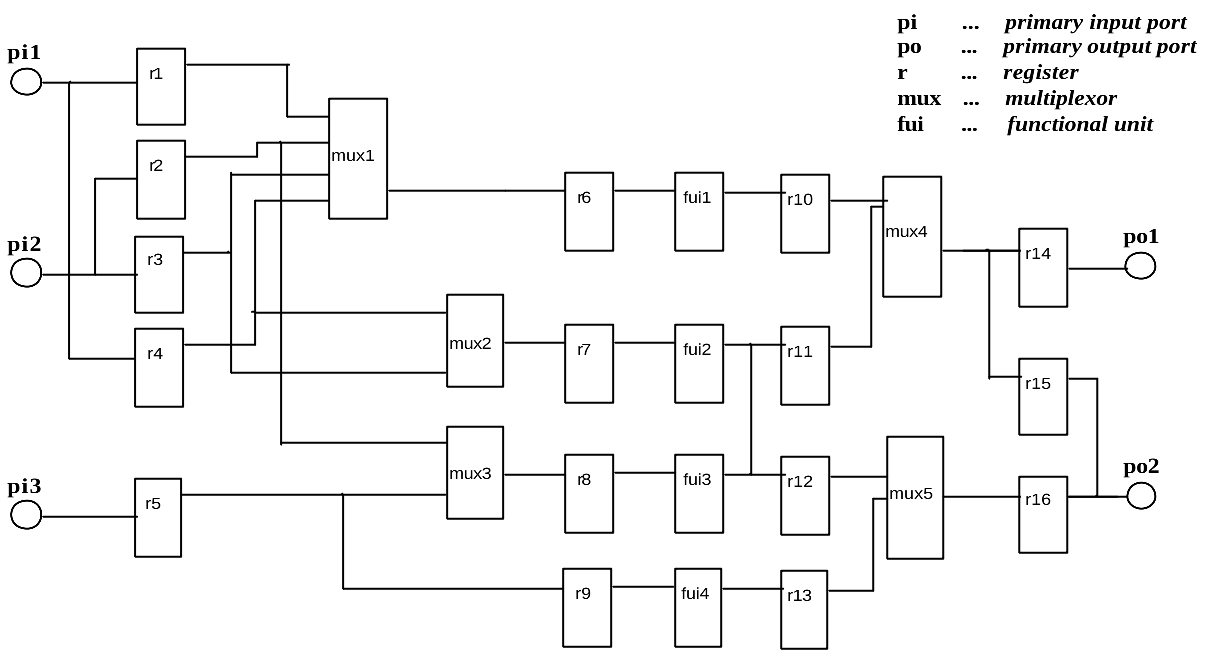

Fig. 2: Example of an RTL circuit

A register may have one of the following roles [4][9] during the test application process (the roles of registers are illustrated in Fig. 1.):

- *Test driver register* (*tdr*) is the register, which feeds test vectors to the inputs of a functional unit (an i path between the *tir* output and the functional unit input exists). In the *tir* the test vectors are either generated or serially scanned in or loaded along i paths from a register or from a primary input,
- *Test receiver* (*trv*) is the register to which the test responses are loaded from the outputs of a functional unit (an i path between the functional unit output and the trv input exists). The test responses are either serially scanned out or transferred along parallel i paths to a primary output,
- *Test input register* (*tir*) is the first register in the parallel i path between a primary input and a functional unit inputs. It may be either a register the input port of which is interconnected with a primary input in parallel or a register which is the last one in the serial i path through which the test vectors for a functional unit are scanned in,
- *Test output register* (*tor*) is the last register in the parallel i path between the output of a functional unit and a primary output. It may be either a register the output port of which is interconnected with a primary output in parallel or a register which is the first one in the serial i path through which the test responses of a functional unit are scanned out.

According to their roles, registers depicted in Fig. 2 are classified into the following sets: test input registers set *TIR = {r1, r2, r3, r4, r5}*, test driver registers *TDR = {r6, r7, r8, r9}*, test receivers *TRV = {r10, r11, r12, r13}* and test output registers *TOR = {r14, r15, r16}*.

3.2 Order Relation

Definition 1: A relation $\leq$ is an (unsharp) order on a set *M* iff $\leq$ it is *reflexive, transitive and antisymetric*. Set *M* with the order $\leq$ on it is called an ordered set and it is denoted as $(M, \leq)$.

In definition 1, relation $\leq$ represents the data flow through the circuit structure. *M* is a set of all circuit elements. In $(M, \leq)$, it is possible to determine whether a data flow is possible between two arbitrary elements *a, b Î M*. Namely, if $a \leq b$, then a data flow is possible (directly or by means of inter-elements) from the output of element *b* to the input of element *a*. Otherwise (if not $(a \leq b)$), a data flow in the direction from *b* to *a* is not possible.

Definition 2: Two elements *a, b* of $(M, \leq)$ are called *comparable* iff $a \leq b$ or $b \leq a$. Otherwise they are called *uncomparable*. Element *a Î A* $\subseteq M$ is called the *least element of A* iff $\forall\ b \in A$: $a \leq b$. Element $a \in A \subseteq M$ is called a *greatest element of A* iff $\forall\ b \in A$: $b \leq a$.

Comparability of two elements reflects the fact that a connection exists between them. Thus, there is no way of transferring a diagnostic data between elements that are uncomparable. Supposing at least one data path exists between primary inputs and primary outputs then 1) circuit elements encountered in this path belong to set *A*, 2) the least element of *A* is a primary input $pi \in PI$ and 3) the greatest element of *A* is primary output $po \in PO$.

Definition 3: The element $a \in M$ is called a *low limit of A* $\subseteq M$ iff $\forall\ b \in A$: $a \leq b$. Element $a \in M$ is called a *high limit of A* $\in M$ iff $\forall\ b \in A$: $b \leq a$.

As the least and greatest elements of A represent the start (first) element and the end (last) element of a certain i path, so by means of low limit and high limit elements, it is possible to construct a set of elements that can be utilized for expansion (further information) of the corresponding i path – low limit element informs about the data-flow successor of the last element in an i path and high limit element informs about the data-flow predecessor of the first element in i path.

Definition 4: Element $b \in M$ *directly covers* element $a \in M$ iff $a \leq b$, $a \neq b$ and for $a \leq c \leq b$: $a = c$ or $b = c$.

If the circuit element a is directly covered by circuit element b, then data inputs of a (covering) circuit element b are directly connected to data outputs of (covered element) a.

Definition 5: Element $b \in M$ *indirectly covers* element $a \in M$ iff $a \leq b$, $a \neq b$ and $\exists\, c \in M$: $a \leq c \leq b$, where $c \neq a$ and $c \neq b$.

If circuit element a is indirectly covered by circuit element b, then data inputs of a (covering) circuit element b are not directly connected to data outputs of (covered element) a – a data path consisting of at least one circuit element exist between the output of a and input of b. In the following text, situation "a is directly covered by b" is denoted as $a <= b$ and situation "a is indirectly covered by b" is denoted as $a <<= b$. Alike, situation "a is not directly covered by b" is denoted as $a <\neq b$ and situation "a is not indirectly covered by b" is denoted as $a <<\neq b$.

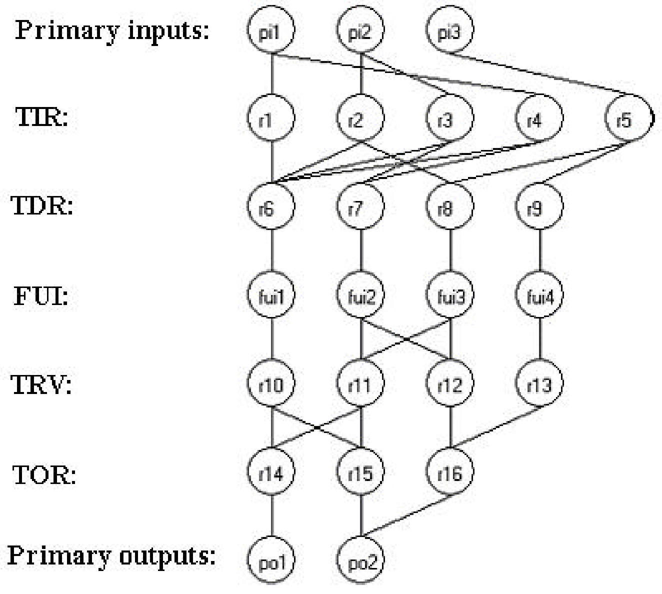

Fig. 3: Hasse diagram of a data-flow for Fig. 2

3.3 Hasse Diagram

A visual representation of an ordered set $(M, \leq)$ via the cover relation is called a *Hasse diagram* (a and b are connected by an edge in such a way that b is placed at higher position than a iff a is covered by b, i.e. a connection between output of b to input of a exists, i.e. data flow between them is possible). If 1) M is a set of circuit elements, 2) those elements are connected according to definitions 1–7 where 3) a data-flow is represented by a cover relation then a data-flow among primary inputs/outputs, tested functional units and selected registers can be graphically represented by a Hasse diagram of a data-flow. The diagram is defined as a 2-tuple $G_H = (M, E)$, where

- M … is a set of circuit elements, M = PI $\cup$ PO $\cup$ TIR $\cup$ TDR $\cup$ FUI $\cup$ TRV $\cup$ TOR
- E … is a set of edges (a subset of a direct coverage relation, see definitions 5 and 6). An edge can exist only between the following pairs of elements of M (in the following list, an element on the left (right) is a covering (covered) element):
    - $pi \in PI \wedge tir \in TIR$
    - $tir \in TIR \wedge tdr \in TDR$
    - $tdr \in TDR \wedge fui \in FUI$
    - $fui \in FUI \wedge trv \in TRV$
    - $trv \in TRV \wedge tor \in TOR$
    - $tor \in TOR \wedge po \in PO$

An edge between any other elements is not allowed.

As an example, Hasse diagram $G_H = (M,E)$ for the circuit from Fig. 2 is depicted in Fig. 3. In G_H, each circuit element is displayed as a vertex of G_H – here registers, primary inputs/outputs and functional units are displayed as circles. Generally, other elements can exist in data-paths between *tir* and *tdr* or between *trv* and *tor.* Because such elements are not important for selecting registers into scan, they are excluded from G_H (which is an abstraction of circuit data-path for the purposes of selecting registers into scan) structure. Thus, if there is an edge between two vertices a, b of G_H, it does not imply a circuit element c does not exists in the data-flow between a and b. In G_H, a data-flow of each i path starts at circuit primary inputs and ends at circuit primary outputs. Because G_H is a Hasse diagram, a covering element (data-flow source) is displayed higher than a covered element (data-flow target) in G_H.

If an element a is covered by only one element b, then element b is called an *indispensable element* (of an element a, see following definition) in given Hasse diagram G_H. Because by removing of *indispensable elements* the assumption will become invalid, *indispensable elements* have to remain in (therefore they cannot be removed from) G_H structure. Next the definitions of various *indispensable elements of Hasse diagram* will be presented.

Definition 6: Element $b \in M$ of G_H is called an *indispensable element* iff $\exists\, a \in M$: $\forall\, c \in M\,(c$ ¹ b a c ¹ $a)$: a is not directly covered by c. The following types of *indispensable elements* elements are distinguished:

i. Register *r*∈*TDR* is called an *indispensable tdr register* if it directly covers *fui*∈*FUI*, which is not directly covered by another *tdr* register, i.e., ∀*q*∈ *TDR (q ¹ r)*: *fui* <≠ *q*.
ii. Register *r*∈*TRV* is called an *indispensable trv register* if it is directly covered by *fui* ∈ *FUI*, which does not directly cover another *trv* register, i.e. ∀ *q*∈ *TRV (q ¹ r)*: *q* <≠ *fui*.
iii. Register *r* ∈ *TIR* is called an *indispensable tir register* if 1) it directly covers register *q*∈*TDR*, 2) *q* is *an indispensable tdr* register and 3) ∀ *p*∈*TIR (p ¹ r)*: *q* <≠ *p*.
iv. Register *r* ∈ *TOR* is called an *indispensable tor register* if it is directly covered by a register *q*∈*TRV*, *q* is *an indispensable trv* register and ∀*p*∈*TOR (p ¹ r)*: *p* <≠ *q*.
v. Primary input *pi*∈*PI* is called an *indispensable primary input* if it directly covers register *r*∈ *TIR*, *r* is *an indispensable tir* register and ∀ *pi'*∈*PI (pi'¹ pi)*: *r* <≠ *pi'*.
vi. Primary input *po*∈*PO* is called an *indispensable primary output* if it is directly covered by register *r*∈*TOR*, *r* is *an indispensable tor* register and ∀ *po'* ∈ *PO (po' ¹ po)*: *po'* <≠ *r*.

The main goal of proposed methodology is to detect minimal (i.e. the most shared) set of *indispensable elements* that can be used for testing all functional units by means of the data-flow through circuit *i paths*. To apply test to each functional unit, exactly one *i path* (with a data flow from primary inputs to primary outputs) will be detected. All such *i paths* are characterized by sharing their sub-paths in the greatest way (i.e. characterized by maximal number of shared registers, that is by minimum number of registers in total). Registers included in those *i paths* will be used in the test application.

3.4 Proposed Algorithms

The search algorithm based on the analysis of Hasse diagram whose objective is the identification minimal number of registers to be included into scan chain consists of six partial tasks. For this purpose, G_H_Minimisation procedure was developed whose input is the non-minimised Hasse diagram reflecting the data flow through the circuit under analysis. It is assumed that the elements of the circuit were subdivided into subsets of *M* set. Every partial task is performed by means of the Coverage_Minimisation(G_H, *List_A*, *List_B*) general algorithm for minimising the coverage between two sets of Hasse diagram. One subset (List_A parameter) contains covering elements while the other one consists of elements, which are supposed to be covered (List_B parameter). As a result of applying the procedure, we gain *Data Flow Hasse Diagram*, which contains functional units *fuis*, primary inputs/outputs and registers which will be included into the *fui* test application process.

Algorithm 1:
The input data structures are Hasse diagram (G_H) and lists of covering (*List_A*) and covered (*List_B*) vertexes of G_H.

Coverage_Minimisation(G_H, *List_A*, *List_B*)
1. Do the identification of *indispensable* elements in *List_A* (the list of covering elements).

2. Create a new *List_B'* containing elements of *List_B* (the list of covered elements) which will contain only the elements with the edge(s) leading to the indispensable elements of the *List_A* (identified in the preceding step).

3. Delete from G_H all the edges leading from elements incorporated in *List_B'* to *List_ A* elements which are not marked as indispensable in the *List_A*.

4. Delete from *List_A* all the elements which do not cover any element from *List_B* (thus, there is no edge between *a* *Î List_A* and *b Î List_B* elements).

5. Are all the elements remaining in *List_A* marked as indispensable? If yes, the algorithm ends – go to step 8, else continue by step 6.

6. Identify in *List_A* all the elements which are not indispensable and insert them into a new auxiliary *List_A'*.

7. By means of the the auxiliary procedure *Decision(…)* select *a' Î List_A'* as the output of the auxiliary procedure (according to the predefined criteria) which will be marked as *indispensable* in *List_A*. Then go to step 2.

8. End of procedure.

End of Algorithm 1.

Algorithm 2:
The input data structure is *List_A': List_A'* ⊆ *List_A*, ∀ *a'* ∈ *List_A'*: a' ∉ Indispensable_A

Decision(*List_A'*):

1. Is there any element *a' Î List A'* marked as „indispensable" in another part of *G_h_minimisation(…)*? If not, then go to step 2 else delete all elements from *List_A'* without this mark.

2. Are there all elements *a' Î List A'* marked as „candidate for scan"? If yes, go to step 3 else delete all elements from *List_A'* without this mark.

3. Search for *a' Î List_A'* which has the maximum number of edges leading to elements from *List_B*. If there is only one *a'* go to step 7 else delete from *List_A'* such elements which have maximum number of edges leading to *List_B* elements lower than the maximum number

4. Search for *a' Î List_A'* which has the lowest node degree. If there is only one *a'* go to step 7 else delete from

List_A' all elements with „the node degree“ value higher than the lowest.

5. Search for *a' Î List_A'* which has the lowest number of *i-paths* leading to primary inputs/outputs? If there is only one *a‘* go to step 7 else delete from *List_A'* all elements with the „number of i-paths“ value higher than the lowest.

6. Is there just one element *a' Î List_A'* from which an i path with the highest sum of edge value leads to PI/PO? If there is only one *a‘* go to step 7 else choose the first element from *List_A'* and mark it as *a'*.

7. Mark the *a*' element as a candidate for indispensability in the *List_A*.

End of Algorithm 2.

Algorithm 3:
The assumption:
M = PI È PO È TIR È TDR È FUI È TRV È TOR
G_H = (*M,E*) …Hasse diagram of diagnostic data flow for *M* set

The G_H_minimisation (G_H)
1. Coverage_Minimisation (G_H, TDR, FUI)
2. Coverage_Minimisation (G_H, TIR, TDR)
3. Coverage_Minimisation (G_H, PI, TIR)
4. Coverage_Minimisation (G_H, TRV, FUI)
5. Coverage_Minimisation (G_H, TOR, TRV)
6. Coverage_Minimisation (G_H, PO, TOR)

End of Algorithm 3.

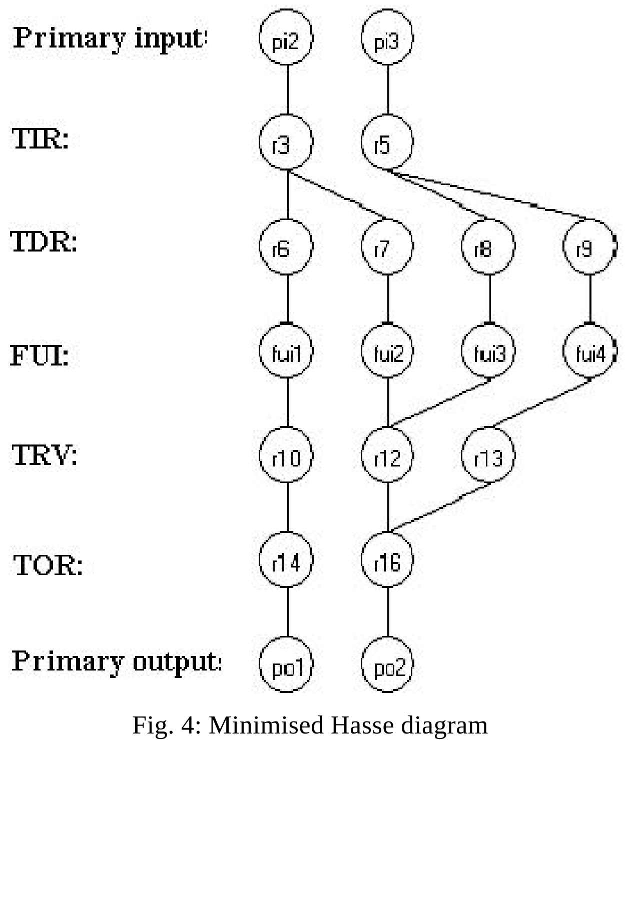

Fig. 4: Minimised Hasse diagram

4 Experimental Results

In Fig. 5, the Hasse diagram of Diffeq benchmark circuit data-flow is shown. The corresponding minimised Hasse diagram of a data-flow is depicted in Fig. 6; registers R1, R4, R5, and R6 are selected to be included into scan chain. Fig. 4 illustrates minimised diagram of a demo exa mple that was depicted in Fig. 2 and Fig. 3. According to Fig. 4, following registers are to be included into scan chain: R3, R5, R6, R7, R8, R9, R10, R11, R13, R15, R16.

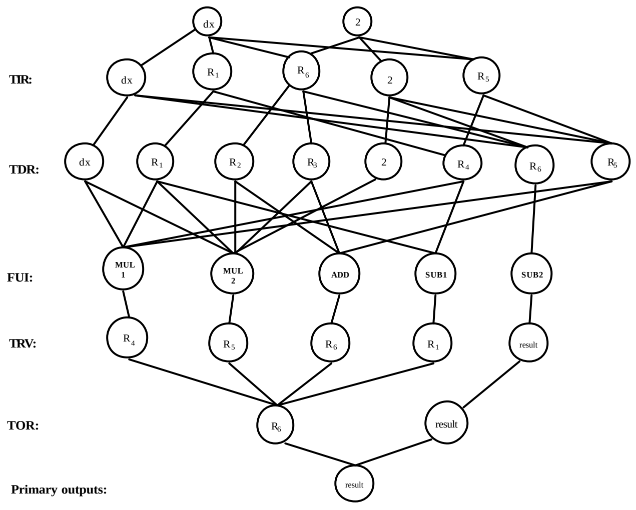

Fig. 5: Hasse diagram of Diffeq data-flow

Other methodologies for selecting registers into scan chain exist. Generally stated, such methodologies are based on different criteria for inserting registers into scan, thus a set of selected registers can differ according to the criteria. Methodologies published in [5], [13], [18] dealt with a selection of registers in the scan chain for Diffeq benchmark circuit too, so we can compare results gained by those methodologies and by the proposed one in short. All following results are related to Diffeq circuit. In [5], registers R4, R6, R1 and R5 were selected to scan chain in order to achieve an optimal trade-off among selected design constraints and Diffeq testability parameters. Selected registers were supposed to be included into one scan chain in the presented order. The criteria were to achieve the highest possible controllability and observability during a random-pattern test generation and to fulfil given design constraints simultaneously. The quality of a selection was evaluated using special-purpose testability measures and a cost/quality trade-off function. In [13], which is based on formal approach for selecting registers into scan, registers R1, R4 and R6 or registers R1, R3 and R4 are supposed to be included into scan. The criteria were to select minimal set of registers in such a way that the number of parallel paths within the Diffeq structure would be maximized (i.e. hopefully: test application time minimized). In [18], criteria for including registers into the scan chain were as follows: 1) to make all global feedback loops winthin Diffeq structure easily controllable (observable) by means of Diffeq primary inputs (primary outputs) and 2) to include registers into the scan chain at the cost of minimal area and primary input/output overheads. The method was based on a utilization of a genetic algorithm. As the result, only registers R1 and R6 were included into scan chain.

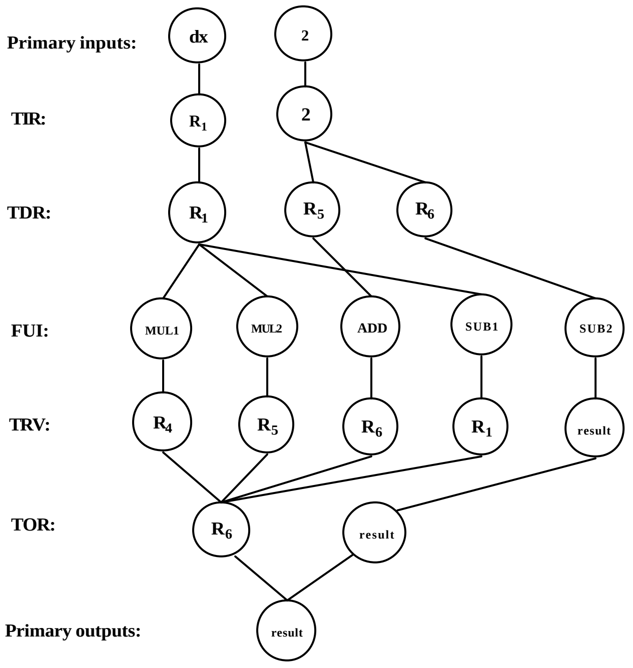

Fig. 6: Minimized Hasse diagram for Diffeq

5 Future Research Perspectives

In our previous research we developed a methodology, which allows to schedule test application of an RTL circuit [3]. It was taking into account the possibility of sharing test resources (i.e. registers, multiplexers and connections) for the test of separate *fuis*. The result of applying the methodology to an RTL structure is the list of test phases with the identification of *fuis*, which can be tested in parallel. We also developed a methodology, which was based on the classification of elements (especially registers) and on their role during test application, which allows develop new types of test application methodologies [9]. We see now that the merging of the methodologies can be further utilised for synchronised test application to an RTL circuit. As the synchronised test application we denote the methodology of selecting registers through which the test will be applied to *fuis* such that the transports of test pattern to inputs of *fuis* will require the same number of system clock pulses as the transport of test responses from outputs of *fuis*. We intend to utilise Hasse diagrams to develop, implement and verify the methodology.

6 Conclusions

In this paper new methodology was presented for selection of circuit register through which test will be applied. The theory of Hasse diagram is a suitable tool for the proposed methodology. The Hasse diagram of a data flow was defined as formal tool, which can depict the flow of diagnostic data trough the circuit under test. The procedure for the minimisation of Hasse diagram of a data-flow was developed and implemented. The proposed procedure can find a minimal number of *tir*, *tdr*, *tor* and *trv* registers and primary ports inside the circuit under test, which are required to implement test controller. The selection of registers is influenced by the predefined criteria. The criteria reflect maximum number of registers shared during the test of functional units. The shared register belongs to *i paths* through which diagnostic data are transported. The selection of registers could be possibly done by another criteria. The methodology was verified on many practical RTL circuits, e.g. Diffeq, Tseng (both from HLSynth92 [19] benchmark suite) or Bert benchmark circuits.

References

[1] ABADIR, M. S. – BREUER, M.: A knowledge based system for designing testable VLSI chips, IEEE Design&Test, August 1985, pp. 56–68

[2] BENABDENBI, M. – MAROUFI, W. – MARZOUKI, M.: CAS-BUS: A Scalable and Reconfigurable Test Access Mechanism for Systems on a Chip, In: Proceedings of Design, Automation and Test in Europe, IEEE Press, 2000, pp. 141–145

[3] BLATNÝ J. – HLAVICKA, J. – KOTÁSEK, Z.: RT level test scheduling, IEEE European Test Conference, Rotterdam, 1993, pp. 499–500

[4] BLATNÝ, J. – KOTÁSEK, Z. RT Level Test Scheduling, In: Computers and Artificial Intelligence, Vol. 16, No. 1, 1997, p. 13–29.

[5] BUKOVJAN, P.: Allocation for Testability in High-Level Synthesis. PhD thesis, Institute National Polytechnique de Grenoble, 2000

[6] CHAKRABARTY, K.: Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming, Trans. CAD of IC and Syst, Vol. 19, No. 10, Oct. 2000, pp. 1163–1174

[7] CHANDRA, A. – CHAKRABARTY, K.: A Unified Approach to Reduce SOC Test Data Volume, Scan Power and Testing Time, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 22, No. 3, March 2003, pp. 352–362

[8] GHOSH S. – BASU S. – TOUBA N. A.: Joint Minimization of Power and Area in Scan Testing by Scan Cell Reordering, In: Proceedings of the IEEE Computer Society Annual Symposium on VLSI, IEEE Computer Society, 2003, pp. 246–249

[9] HLAVICKA, J. – KOTÁSEK, Z. – ZBORIL, F.: Partial Scan Methodology for RTL Designs, In: Compendium of Papers European Test Workshop, Constance, 1999, p. 2

[10] HUANG, Y. – CHENG, W. T. – TSAI, C. C. – MUKHERJEE, N., SAMMAN, O., ZAIDAN, Y. – REDDY, S. M.: On Concurrent Test of Core-Based SOC Design, Journal of Electronic Testing-Theory and Applications, Vol. 18, No. 4-5, August-October, 2002, pp. 401–414

[11] INOUE T. – MIURA T. – TAMURA A. – FUJIWARA H.: A Scheduling Method in High-Level Synthesis for Acyclic Partial Scan Design, In: Proceedings of 11th Asian Test Symposium, IEEE Computer Society, 2002, pp. 128–133

[12] KOTÁSEK, Z. – MIKA, D. – STRNADEL, J.: Test scheduling for embedded systems, In: Proc.

EUROMICRO Symposium on DSD–Architectures, Methods and Tools, ICSP, 2003, pp. 463–467

[13] KOTASEK, Z. – RUZICKA, R. – HLAVICKA, J.: Formal Approach to the RTL Testability Analysis. In: Proceedings of 1st IEEE Latin America Test Workshop, 2000, pp. 256–261

[14] LARSSON, E. – ARVIDSSON, K. – FUJIWARA, H. – PENG, Z.: Integrated Test Scheduling, Test Parallelization and TAMDesign, In: Proceedings of 11th Asian Test Symposium, IEEE Computer Society, 2002, pp. 397–404

[15] LARSSON, E. – PENG, Z. B.: An Integrated Framework for the Design and Optimization of SOC Test Solutions, Journal of Electronic Testing-Theory and Applications, Vol. 18, No. 4-5, August-October, 2002, pp. 385–400

[16] MARINISSEN, E. J. – KAPUR, R. – LOUSBERG, M. – MCLAURIN, T. – RICCHETTI M. – ZORIAN, Y.: On IEEE P1500's Standard for Embedded Core Test, Journal of Electronic Testing-Theory and Applications, Vol. 18, No. 4-5, August-October, 2002, pp. 365–383

[17] POUGET, J. – LARSSON, E. – PENG, Z. – FLOTTES, M. – ROUZEYRE, B: An Efficient Approach to SoC Wrapper Design, TAM Configuration and Test Scheduling, In: Proceedings of 8th IEEE European Test Workshop, IEEE Computer Society, 2003, pp. 51–56

[18] STRNADEL, J. – KOTÁSEK, Z. – MIKA, D.: Methodologies of RTL Partial Scan Analysis and Their Comparison, In: Proceeding of IEEE Workshop on Design and Diagnostic of Electronic Circuits and Systems, Poznan, PL, UNI-DRUK, 2003, p. 233–238

[19] http://www.cbl.ncsu.edu/pub/Benchmark_dirs/HLSynth92