Non-Scan Design-For-Testability of RT-Level Data Paths

Sujit Dey and Miodrag Potkonjak
C&C Research Laboratories, NEC USA
Princeton, NJ 08540

Abstract - *This paper presents a* non-scan *design-for-testability technique applicable to register-transfer (RT) level data path circuits, which are usually very hard-to-test due to the presence of complex loop structures. We develop a new testability measure, and utilize the RT-level structure of the data path, for cost-effective re-design of the circuit to make it easily testable, without having to either scan any flip-flop or break loops directly. The non-scan DFT technique was applied to several data path circuits. Experimental results demonstrate the feasibility of producing non-scan testable data paths, which can be tested* at-speed*. The hardware overhead and the test application time required for the non-scan designs is significantly lower than the corresponding partial scan designs.*

I. Introduction

Several high level synthesis for testability approaches have been proposed to generate easily testable data paths for both Built-In-Self-Test (BIST)-based testing methodology [1, 2, 3], and Automatic Test Pattern Generation (ATPG) methods [4, 5, 6, 7]. However, almost all BIST-based approaches assume a scan design methodology since random testing is not well-suited for sequential circuits. Also, almost all the ATPG-based high level synthesis for testability approaches assume the use of scan registers to make the data paths testable. Like the high-level techniques, all the existing Register Transfer (RT) level techniques [8, 9, 10] are scan-based, and cannot generate testable data paths without the use of scan.

Scan-based techniques have the disadvantage that the test application time is very large compared to non-scan designs, since the test vectors have to be shifted through the scan chain. Reduction of test application time has been addressed in several ways, like arranging scan flip-flops in parallel scan chains [11], and reconfiguring scan chains [12]. While the former method is limited by the number of primary inputs and primary outputs of the circuit, the latter approach is limited by the ability of the circuit to be decomposed into a set of kernels. On the other hand, non-scan DFT techniques do not require to scan any FFs, thus eliminating the need to shift test vectors through scan chains, and greatly reducing the test application time.

However, the biggest disadvantage of scan-based DFT techniques is the inability of scan designs to be tested at-speed, which assumes significance in light of recent studies showing that a stuck-at test set applied *at-speed* identifies more defective chips than a test set having the same fault coverage but applied at a lower speed [13]. The studies motivated researchers to investigate non-scan DFT techniques to make sequential circuits testable by introducing controllability and observability points [14]. The main advantage of the non-scan designs is that the test vectors can be applied at-speed.

This paper presents a new DFT technique to make data paths testable, without using scan registers. We assume that the control signals to the data path can be made fully controllable by loading the FFs of the controller with primary input signals, using the technique in [14]. Knowledge of the RT-level structure, as well as the functions of the RT-level components, are utilized in the proposed DFT approach. Instead of conventional techniques of selecting flip-flops (FF) to make controllable/observable, outputs of execution units (EXU) are selected using the EXU S-graph introduced in the paper.

We introduce a new testability measure, k-level controllable and observable loops, and demonstrate that it suffices to make all the loops k-level controllable/observable, $k > 0$, to achieve very high test efficiency. The new testability measure eliminates the need by traditional DFT techniques to make all loops directly (0-level) controllable/observable, reducing significantly the hardware overhead required, and making the non-scan DFT approach feasible and effective. We introduce the use of dual points to make one loop controllable while making another loop observable. We present efficient algorithms to add the minimal hardware possible to make all loops in the data path k-level controllable/observable, without the use of scan FFs.

We applied our non-scan DFT technique to several moderately sized data paths. The experimental results demonstrate the effectiveness of the k-level heuristic and our non-scan DFT technique to design highly testable data paths, with nominal hardware overhead. Besides the main advantage of at-speed testing, experimental results also demonstrate that the hardware overhead and the test application time required for the non-scan designs is significantly lower than the partial scan designs.

II. Scan and Non-Scan DFT of RT-level Data Paths: An Illustration

Figure 1(a) shows the register-transfer (RT) level data path for 4th order IIR cascade filter, synthesized from behavioral description using the HYPER high-level synthesis system [15]. The corresponding register S-Graph [16, 17, 6] in Figure 1(b) shows the dependencies between the registers of the data path. The register S-graph reveals the existence of several loops involving the registers. As can be expected, sequential ATPG is very difficult for the data path, as indicated in Table 2 by the row *Orig*.

The testability of the data path can be improved using partial scan techniques [16, 17, 18] to break all the loops of the circuit. Since the MFVS of the S-graph in Figure 1(b) is 3, breaking all the loops needs scanning at least 3 registers, namely LA1, LA2, and LM1. For the 20-bit IIR filter data path shown in Figure 1(a), 60 scan FFs are needed by the partial scan tool OPUS [18], and Lee-Reddy's partial scan tool [17], shown by the rows *Opus* and *LR* respectively in Table 2. The sequential ATPG program HITEC [19], can achieve 100%

Figure 1: Characteristics of a design implementing the 4th order IIR cascade filter: (a)the RT-level data path, (b) Register S-graph, (c) EXU S-graph

test efficiency on the scan designs. Besides the high area overhead, the scan designs have high test application time, indicated by column *Tappl*. Also, the scan designs cannot be tested at-speed.

A. EXU S-Graph

Unlike a partial scan DFT technique, it is not necessary for a non-scan DFT technique to restrict to only registers the choice of nodes to break, that is, make controllable/observable. Also, as opposed to making the same point controllable as well as observable, it may be more cost-effective to make some points controllable, while some other points observable. In this section, we introduce the *EXU S-graph*. We show that in a data path, (the outputs of) EXUs are better choices for controllable/observable points than registers. Each node in the EXU S-graph represents an EXU in the data path. There exists a directed edge from node u to node v, labeled i, and denoted $u \xrightarrow{i} v$, if there exists a direct path from EXU u to the ith register file of EXU v, without going through any other register.

The EXU S-graph for the data path in Figure 1(a) is shown in Figure 1(c). The EXU S-graph has several loops. There are two loops in the EXU S-graph between M1, A2 and TU1, namely $M1 \xrightarrow{1} A2 \xrightarrow{1} TU1 \xrightarrow{1} M1$, and $M1 \xrightarrow{2} A2 \xrightarrow{1} TU1 \xrightarrow{1} M1$. However, all the loops in the EXU S-graph pass through the two EXUs, A1 and A2. Hence, the MFVS of the EXU S-graph is 2, as opposed to an MFVS of 3 for the register S-graph of Figure 1(b).

B. Non-Scan DFT of the IIR Cascade Filter

Next, we proceed with the task of non-scan DFT of the data path in Figure 1(a). Since A1 and A2 form the MFVS of the EXU S-graph, making the outputs of A1 and A2 controllable/observable would break all the loops directly, that is, make all the loops *0-level* controllable/observable. That is, any value at the outputs of A1, A2 can be controlled and observed in 1 clock cycle (time frame). Compared to the Register S-graph solution, which requires making three registers controllable/observable, this solution seems better. However, we show that much less expensive non-scan DFT techniques would suffice to make the data path testable.

The EXU S-graph in Figure 1(c) reveals that all loops through A2 are observable, since A2 goes directly to the PO Out. Hence, we need to add only a controllability point to output of A2, while adding both a controllability and observability point to the output of A1. Figure 2(a) shows the modified data path of Figure 1(a), with test hardware added (shown in bold) to insert one controllability point at the output of A1 and A2, and one observability point from the output of A2. The output of A1 is made controllable by multiplexing it with the PI *In*. The multiplexor is controlled by the test pin *ntest*, which is set to "0" during the normal operation of the data path, and can be set to any value required during the test mode. Similarly, the output of A1 is made observable by multiplexing A1 with a PO *Out*, as shown in Figure 2(a). A test efficiency of 100% could be achieved on the resultant data path, as evidenced by the row *0-lev* in Table 2. The test hardware overhead required for the modified data path is 429 cells, (5.7% of the original data path), which is less than the overhead of 665 cells needed for the scan designs (rows *Opus*, *LR* in Table 2). Besides having the main advantage of at-speed testing, the number of clock cycles required for test application (column *Tappl*) for the non-scan design is much less than the scan design.

It is not necessary to make the loops of the data path directly (0-level) controllable/observable. Figure 2(b) shows an alternate testable design, with the non-scan test hardware shown in bold. Instead of adding a controllability point to the output of A2, only a constant ("0", the identity element of the adder) is added to the right register file (RA2) of A2. Any value at the output of A2 can still be justified by at most two time frames. For example, if a value of 9 needs to be justified at the output of A2, in one time frame the registers LA2 and RA2 can be set to appropriate values 9 and 0, and in the next time frame the values of LA2 and RA2 can be justified by *In* and the constant. Adding the constant requires much less hardware overhead than adding a controllability point at the output of A2, since the multiplexor logic associated with constant signals can be pruned. The loops through A2 are now 1-level controllable. The resultant (1-level controllable/observable) data path shown in Figure 2(b) has less hardware overhead than the 0-level solution shown in Figure 2(a). Also, a high test efficiency of 99% could be achieved on the resultant data path, as evidenced by the row *1-lev* in Table 2.

The data path in Figure 2(c) demonstrates more effectively the benefits of non-scan DFT at the RT-level, and the new notion of k-level controllable/observable loops. The data path shows the addition of just two constants to the right registers, RA1 and RA2, of the EXUs A1 and A2 respectively. As explained in section III, all the loops in

Figure 2: **Non-scan** design-for-testability of the data path in Figure 1(a), (a) 0-level testable data path: all loops are 0-level controllable/observable, (b) 1-level testable data path, (c) 2-level testable data path

the EXU S-graph now become 2-level or less controllable/observable. The test hardware required is significantly less than the 0-level and 1-level testable data paths shown in Figures 2(a) and 2(b) respectively. The area overhead is only 120 cells (row *2-lev* Table 2), as compared to an overhead of 665 cells for the scan designs, 429 cells for the 0-level non-scan design, and 349 cells for the 1-level non-scan design. The 2-level testable design, however, has a very high test efficiency of 98%, comparable with the test efficiency achieved by the more expensive scan designs, and the 0-level and 1-level non-scan designs.

The non-scan designs and their high test efficiency results demonstrate the feasibility of using non-scan DFT schemes, and establishes the technique of making loops k-level controllable/observable as an efficient alternative to the traditional DFT technique of breaking all loops directly. The new testability metric is explained in the next section.

III. K-Level Controllable/Observable Loops: A Cost-Effective DFT Approach

We first define k-level controllable/observable nodes. We consider EXUs (their output bus) as the only possible nodes. We introduce non-scan DFT techniques to make nodes k-level controllable/observable. Next, we define k-level controllable/observable loops in terms of k-level controllable/observable nodes.

Definition 1 *An EXU M is k-level controllable/observable if any value on the output of M can be justified/propagated in at most k+1 clock cycles (time frames). Alternatively, for any value that needs to be justified at the output of M, there exists at least one vector sequence of length at most k+1 that justifies the value.*

Consider the data path shown in Figure 2(c). The output of A1 is 2-level controllable, as explained below. For example, to justify a value of 15 at the output of A1, in the first time frame LA1 can be set to 15, and RA1 to 0. In the second time frame, the value of RA1 can be immediately justified by the constant. To justify the value of LA1, which is the output of A2, the input registers of A2, LA2 and RA2, are set to 15 and 0 respectively. In the third time frame, RA2 can be justified because of the presence of the constant to RA2. Suppose the constant K4 to M3 is 1. LA2 can be justified by setting In to 15. Similarly, any value at the output of A1 can be justified in 3 time frames, making A1 2-level controllable. Note that without the addition of the constants, the output of A1 is not controllable, as is in the original data path of Figure 2(a). It can be similarly shown that the output of A1 is 2-level observable, since any value at the output of A1 can be propagated out in 3 clock cycles.

The output of an EXU, Z, can be made k-level controllable/observable either by a direct scheme, or a register-file based scheme. In the direct scheme, the EXU output is directly multiplexed with a k-level controllable node to make Z k-level controllable. The EXU output is made k-level observable by directly multiplexing it with another node which is k-level observable. Consider the EXU shown in Figure 3(a). Figure 3(b) shows how ALU1 is made k-level controllable and observable using the direct scheme.

In the register-file based scheme, an EXU (output) is k-level controllable if at least one register of each register file of the EXU has a k-1 level controllable input, as shown in Figure 3(c). An EXU is k-level observable if it has an interconnect to a a register file of another EXU, which is k-1 level observable, and whose other register file has a 1-level controllable input. Figure 3(d) shows how ALU1 is made k-level observable.

Definition 2 *A loop is k-level controllable if there is at least one node in the loop which is k-level controllable. A loop is k-level observable if there is at least one node in the loop which is k-level observable.*

Definition 3 *A data path is k-level testable if all loops in the data path are k-level or less controllable and observable.*

Consider the data path shown in Figure 2(c). It has been derived from data path in Figure 1(a), by adding two constants ("0") to the right registers, RA2 and RA1, of the EXUs A2 and A1 respectively. All loops going through A1 are 2-level controllable and 2-level observable since A1 is 2-level controllable/observable, as shown before. Similarly, all loops going through A2 are 1-level controllable/observable. Hence, the data path shown in Figure 2(c) is 2-level testable.

Figure 3: k-level controllability/observability: (a) an EXU, (b) direct scheme for making ALU1 k-level controllable/observable (c) register-file based scheme for making ALU1 k-level controllable, (d) register-file based scheme for making ALU1 k-level observable

IV. Dual Points To Optimize Test Hardware

This section introduces dual points as a powerful technique to optimize non-scan test hardware. A controllability point primarily enhances the controllability of a loop. An observability point primarily enhances the observability of a loop. However, a dual point is used for the dual purpose of enhancing the controllability of one loop, while enhancing the observability of another loop.

Definition 4 *Let us assume that a loop L1 is k_1-level controllable, and another loop L2 is k_2-level observable. A dual point involves multiplexing the output of an EXU in loop L1 with either an input register (register-file based scheme) or the output (direct scheme) of another EXU in loop L2. The dual point simultaneously enhances the observability of loop L1 to $k_2 + 1$ (k_2 for direct scheme), and the controllability of loop L2 to $k_1 + 1$ (k_1 for direct scheme).*

Consider the data path of the 4th order IIR parallel filter shown in Figure 4(a). The original data path is very untestable, as shown by the results of running HITEC (row *Orig*) in Table 5. A non-scan 0-level testable design, using three controllability points and two observability points, is shown in Figure 4(b). The test hardware added is shown in bold. The non-scan design has a very high test efficiency, as evidenced by the row *0-lev* in Table 5.

However, using dual points reduces the test hardware requirement. Adding a constant to the left register of 1+ makes all loops through 1+ 1-level controllable. A dual point added from 1+ to the left register of 3+, and another dual point added from 3+ to the right register of 6+ (with a constant added to the left register of 6+) makes the loops through 3+ 2-level controllable and 2-level observable, the loops through 1+ 3-level observable, and the loops through 6+ 3-level controllable. The resulting data path, shown in Figure 4(c), is 3-level testable. The test hardware added is shown in bold. Note that the hardware overhead for a dual point is the same as a controllability or an observability point. Hence, the dual point solution is less expensive than the the 0-level solution shown in Figure 4(b), which employs controllability and observability points. In fact, the hardware overhead of the dual point solution (row *3-lev*) is 40% less than the overhead of the 0-level solution, as shown in Table 5. Also, the dual point solution has a very high test efficiency, 99%, shown by row *3-lev* in Table 5.

V. Algorithms To Add Test Hardware For K-Level Testable Data Paths

We briefly describe the algorithm which adds the minimal hardware possible to make all loops in the data path k-level controllable and k-level observable, for a user-specified value of k. Since addition of a controllability point (cp) or observability point (op) requires a new interconnect and a multiplexor, we consider that it is always preferable to add constants as a means of enhancing observability and controllability than to add either a cp or an op. Since the number of loops in the EXU S-graph can be exponential, it is not possible to enumerate them individually. Instead, at each step of the algorithm, we count the number of nodes in all loops (strongly connected components) which either have the level of controllability or the level of observability higher than required. Note that all nodes in the EXU S-graph have to be considered for addition of cp or op, not only the nodes in strongly connected components, as is the case when minimum feedback vertex set has to be found.

The input to the algorithm is the target datapath, and the maximum number of allowed cp or op, specified by the user. The following pseudo code summarizes the heuristic algorithm used. A test point, p, refers to either a controllability point or an observability point.

add_test_points()

```
1.  while ∃ a loop whose controllability/observability level > k
2.      if (there is still an available test point) {
3.          for each vertex in S-graph
4.              E(p) ← evaluate test point(p), ∀ test points;
5.          select test point with highest E(p);
6.          add best test point;
        }
7.      else if (there exists a register file without a constant) {
8.          for each vertex
9.              E(p) ← evaluate constant(p);
10.         select constant with highest E(p);
11.         add best constant;
        }
12.     else { request more test points; EXIT; }
13.     update_the_number_of_nodes_in_remaining_SCC();
14. }
```

Both test points and constants are evaluated according to the objective function E(p), where p is the test point or the constant being evaluated: $E(p) = \Delta(LCM(p)) + \Delta(LOM(p))$. The LCM (Loop Controllability Measure) cost is equal to the number of nodes which are in loops whose controllability level is greater than k. Similarly, the LOM (loop observability measure) cost is equal to the number of nodes which are in loops whose observability level is greater than k. Δ denotes the change in the LCM and LOM cost due to insertion of the candidate test point or constant. Details of the algorithm, including calculating the controllability/observability levels of loops, and using dual points, can be found in [20].

Figure 4: Use of Dual Points to optimize test hardware: (a)RT-level data path of the 4th order IIR parallel filter, (b) 0-level testable design using 3 controllability points, 2 observability points, and 5 interconnects, (c) 3-level testable design using 2 dual points, 2 constants, and 2 interconnects

VI. Experimental Results

We applied the non-scan design-for-testability algorithm on several data path circuits, synthesized using the high level synthesis system HYPER [15] from behavioral descriptions [21]. In this section, we report the results obtained on the following data paths: (1) 4th order IIR cascade filter (4IIRcas), (2) 5th order elliptical wave digital filter (EWF), (3) 5th order elliptical wave digital filter, synthesized using high hardware sharing (EWFhigh), and (4) 4th order IIR parallel filter, synthesized using no hardware sharing.

Table 1 shows various parameters of the circuits: the word size of the designs (*Bits*), the number of adders (*Add*), multipliers (*Mult*), registers (*Reg*), multiplexers (*Mux*), and interconnects (*Inter*). The number of cells needed for the final technology mapped circuit, using the SIS technology mapper [22], and the *lib2.genlib* standard cell SCMOS 2.0 library [23], is reported in column *Area*.

The results of applying partial scan and non-scan DFT techniques on the data paths in Table 1 are reported in the Tables 2, 3, 4, and 5. The row *Orig* shows the original design. The rows *Opus* and *LR* show the partial scan designs obtained by using OPUS [18], and Lee-Reddy's tool LR [17]. The designs produced by the non-scan DFT technique presented in this paper are shown in the subsequent rows, with the rows *2-lev*, *1-lev*, *0-lev*, etc. representing the 2-level, 1-level, and 0-level testable designs respectively.

Design	Bits	Add	Mult	Reg	Mux	Inter	Area (Cells)
4IIRcas	20	2	3	12	12	20	7486
EWF	16	3	3	23	29	20	6968
EWFhigh	20	1	1	18	23	6	7538
4IIRpar	20	6	6	23	0	23	9757

Table 1: Characteristics of the RT-Level Data Paths

Type	Test Hardware	Cost (Cells)	Faults	TE%	Tgen (secs)	Tappl (cycles)
Orig	None	0	10004	3.00	23884	-NA-
Partial-Scan DFT						
Opus	60 scan FFs	665	10004	99.97	226	9360
LR	60 scan FFs	665	10004	99.97	210	10080
Non-Scan DFT						
2-lev	2c	120	10086	97.63	6982	109
1-lev	1c,1cp,1op	349	10334	99.32	686	149
0-lev	2cp,1op	429	10448	99.95	292	268

Table 2: 4IIRcas: Cost and Effect of several DFT schemes

For each of the designs, column *Test Hardware* summarizes the test hardware that had to be added to the original design to make the circuit testable. In the case of partial scan designs, the number of scan FFs needed is reported. In the case of each non-scan design, the number of constants *c*, the number of control points *cp*, the number of observability points *op*, or the number of dual points *dp* that had to be added to the original design is shown. For each scan and non-scan design, the test hardware overhead, in terms of extra cells used, is shown in the column *Cost*. The overhead for scan-based designs includes the extra cost of each scan FF, compared to a normal FF, the cost to generate the scan clock, and the cost of buffers needed to distribute the scan clock to the scan FFs.

The testability of the designs was evaluated using the sequential test pattern generator HITEC [19]. For each design, the total number of faults (*Faults*), the percentage test efficiency (*TE%*), that is the percentage of faults either for which a test could be found or which could be identified as untestable, and the test generation time taken (*Tgen*) in

Type	Test Hardware	Cost (Cells)	Faults	TE%	Tgen (secs)	Tappl (cycles)
Orig	None	0	9088	2.00	27423	-NA-
Partial-Scan DFT						
Opus	240 scan FFs	2645	9088	100.00	209	28320
LR	240 scan FFs	2645	9088	99.90	223	32880
Non-Scan DFT						
3-lev	3c	96	9186	54.43	18043	52
2-lev	1c,1cp,1op	256	9380	90.50	4668	72
1-lev	1cp,1op	224	9346	98.09	985	253
0-lev	3cp,3op	671	9798	96.97	1222	278

Table 3: EWF: Cost and Effect of several DFT schemes

Type	Test Hardware	Cost (Cells)	Faults	TE%	Tgen (secs)	Tappl (cycles)
Orig	None	0	9375	2.00	29220	-NA-
Partial-Scan DFT						
Opus	280 scan FFs	3085	9375	99.97	240	45920
LR	280 scan FFs	3085	9375	99.96	251	53760
Non-Scan DFT						
1-lev	1c	20	9397	97.11	2625	218
1-lev	1c,1op	161	9519	98.63	992	297
0-lev	1cp,1op	282	9657	99.79	250	347

Table 4: EWF with high hardware sharing (EWFhigh): Cost and Effect of several DFT schemes

Type	Test Hardware	Cost (Cells)	Faults	TE%	Tgen (secs)	Tappl (cycles)
Orig	None	0	14215	4.00	77527	-NA-
Partial-Scan DFT						
Opus	60 scan FFs	665	14215	99.96	913	5880
LR	60 scan FFs	665	14215	99.97	399	6480
Non-Scan DFT						
3-lev	2c,2dp	347	14497	98.99	13919	151
0-lev	3cp,2op	565	14953	99.91	602	190

Table 5: 4IIRpar: Cost and Effect of several DFT schemes

CPU secs on a Sparc2, are reported. Lastly, the test application time, that is the number of clock cycles needed to apply the test vectors to the design, calculated as (# of test vectors)*(# scan FFs +1), is shown in the column *Tappl*.

Tables 2, 3, 4, and 5 show the ability of the non-scan DFT techniques to make the highly untestable data paths very easily testable, with a significantly smaller test hardware overhead and test application time than the scan designs. Consider the results for the data path circuit EWFhigh, shown in Table 4. The original data path has a very low test efficiency (2%). The scan designs achieve very high test efficiency, but the test hardware overhead is very high (3085 cells), and the test application time needed is 45920 clock cycles. On the other hand, using the non-scan DFT technique presented, the 1-level testable data path uses only 20 extra cells while still achieving a very high test efficiency of 97%. The 0-level non-scan solution achieves almost a 100% test efficiency, while requiring only 282 extra cells, as compared to 3085 required by the scan designs. Also, only 347 clock cycles are needed to apply the test vectors to the 0-level design as compared to 45920 clock cycles needed for the scan implementation.

The experimental results also validate the effectiveness of the k-level controllable/observable loops measure introduced in this paper. The results show that it is not needed to make all the loops directly (0-level) controllable/observable to achieve high test efficiency, as evidenced by the very high test efficiency reported for the k-level testable data paths, $k > 0$. Most significantly, the experimental results demonstrate the feasibility of producing highly testable data paths, which can be tested at-speed, without the use of scan.

VII. Conclusions

We have proposed a new design-for-testability technique to make RT-level data paths testable. As opposed to the earlier DFT techniques for data paths, the proposed technique does not use scan, and the resultant designs can be tested at-speed, enhancing detection of defective chips. The effectiveness of the proposed DFT techniques is largely due to the effectiveness of a new testability measure introduced in this paper, which eliminates the need to break loops explicitly. Experimental results demonstrate the feasibility of generating data paths with high test efficiency without using scan, and with significantly lower test area overhead and test application time than the corresponding partial scan designs.

ACKNOWLEDGEMENT: We wish to thank Vijay Gangaram for his help in the project.

References

[1] S. S. K. Chiu and C. Papachristou. A Built-In Self-Testing Approach For Minimizing Hardware Overhead. In *Proceedings of the International Conference on Computer Design*, 1991.

[2] L. Avra. Allocation and Assignment in High-Level Synthesis for Self-Testable Data Paths. In *Proceedings of the International Test Conference*, 1991.

[3] L. Avra and E. McCluskey. Synthesizing for Scan Dependence in Built-In Self-Testable Designs. In *Proc. ITC*, pages 734 – 743, October 1993.

[4] C.-H. Chen and D. G. Saab. Behavioral Synthesis for Testability. In *Proc. ICCAD*, pages 612 – 615, November 1992.

[5] T. C. Lee, N. K. Jha, and W. H. Wolf. Behavioral Synthesis of Highly Testable Data Paths under Non-Scan and Partial Scan Environments. In *Proc. Design Automation Conf.*, pages 292–297, 1993.

[6] S. Dey, M. Potkonjak, and R. Roy. Exploiting Hardware Sharing in High Level Synthesis for Partial Scan Optimization. In *Proceedings of the International Conference on Computer-Aided Design*, pages 20 – 25, November 1993.

[7] S. Dey and M. Potkonjak. Transforming Behavioral Specifications to Facilitate Synthesis of Testable Designs. In *Proc. ITC*, October 1994.

[8] S. Bhattacharya, F. Brglez, and S. Dey. Transformations and Resynthesis for Testability of RT-Level Control-Data Path Specifications. *IEEE Transactions on VLSI Systems*, 1(3):304–318, September 1993.

[9] V. Chickermane, J. Lee, and J. H. Patel. A Comparative Study of Design for Testability Methods Using High-Level and Gate-Level Descriptions. In *Proc. ICCAD*, pages 620 – 624, November 1992.

[10] H. Harmanani and C. Papachristou. An Improved Method for RTL Synthesis with Testability Tradeoffs. In *Proc. ICCAD*, pages 30 – 35, November 1993.

[11] S.M. Reddy and R. Dandapani. Scan Design Using Standard Flip-Flops. *IEEE Design and Test*, pages 52 – 54, Feb 1987.

[12] S. Narayanan and M.A. Breuer. Reconfigurable Scan Chains: A Novel Approach To Reduce Test Application Time. In *Proc. of ICCAD*, pages 710 – 715, 1993.

[13] P.C. Maxwell, R.C. Aitken, V. Johansen, and I. Chiang. The Effect of Different Test Sets On Quality Level Prediction: When is 80% Better Than 90%? In *Proc. of the Intl Test Conference*, pages 358 – 364, Oct 1991.

[14] V. Chickermane, E.M. Rudnick, P. Banerjee, and J. H. Patel. Non-Scan Design-for-Testability Techniques for Sequential Circuits. In *Proc. Design Automation Conf.*, pages 236 – 241, June 1993.

[15] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak. Fast Prototyping of Data Path Intensive Architectures. *IEEE Design and Test*, pages 40 – 51, 1991.

[16] K.T. Cheng and V.D. Agrawal. A Partial Scan Method for Sequential Circuits with Feedback. *IEEE Transactions on Computers*, 39(4):544 – 548, April 1990.

[17] D.H. Lee and S.M. Reddy. On Determining Scan Flip-Flops in Partial-Scan Designs. In *Proceedings of the International Conference on Computer-Aided Design*, pages 322 – 325, November 1990.

[18] V. Chickermane and J. H. Patel. A Fault Oriented Partial Scan Design Approach. In *Proceedings of the International Conference on Computer-Aided Design*, pages 400 – 403, November 1991.

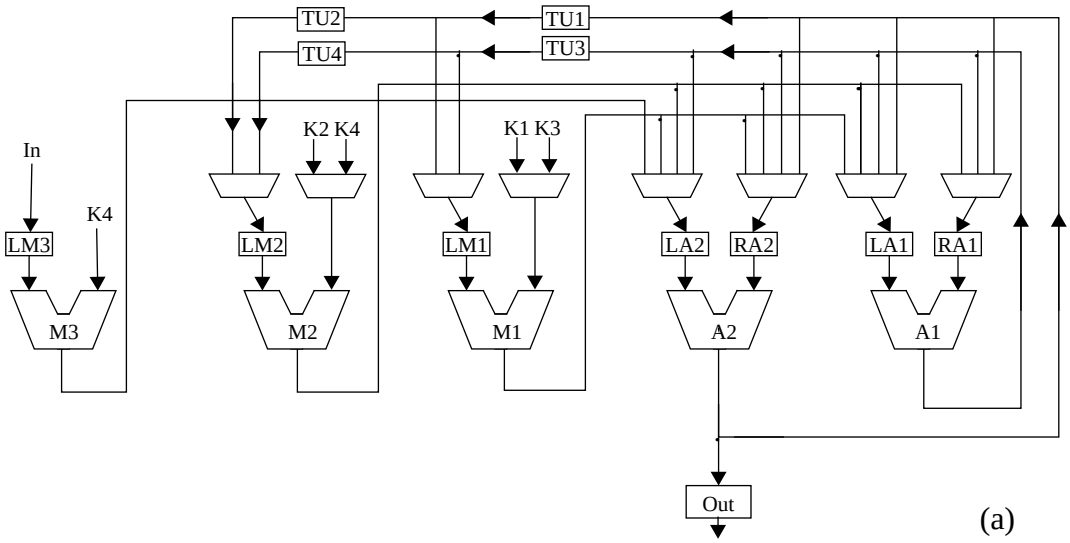
[19] T. M. Niermann and J. H. Patel . HITEC: A Test Generation Package for Sequential Circuits. In *Proc. EDAC*, pages 214–218, 1991.

[20] S. Dey and M. Potkonjak. Non-Scan Design-for-Testability of RT-Level Data Paths. Technical Report 94-C008, C&C Research Labs, NEC USA, May 1994.

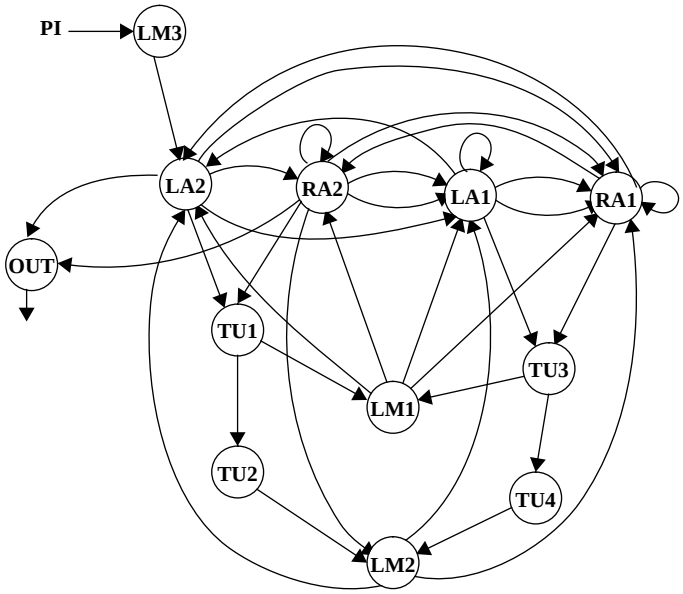
[21] R.A. Haddad and T.W. Parsons. *Digital Signal Processing: Theory, Applications and Hardware*. Computer Science Press, New York, NY, 1991.

[22] E.M. Sentovich, K.J. Singh, C. Moon, H. Savoj, R.K. Brayton, and A. Sangiovanni-Vincentelli. Sequential Circuit Design using Synthesis and Optimization. In *Proceedings of the International Conference on Computer Design*, October 1992.

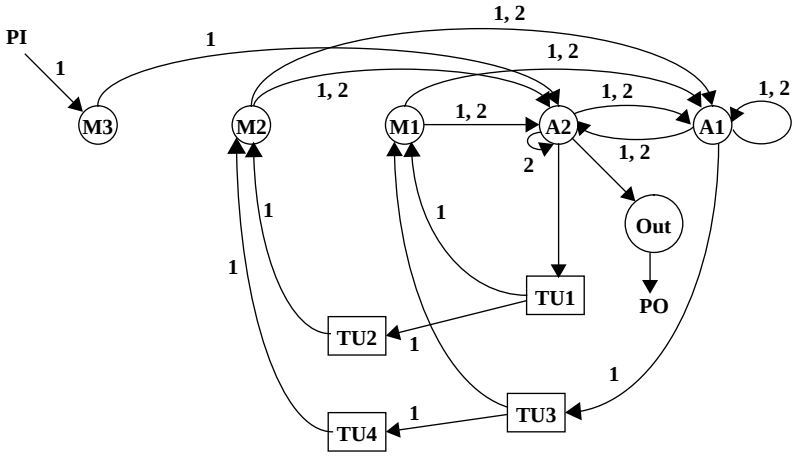
[23] S. Yang. Logic Synthesis and Optimization Benchmarks, User Guide Version 3.0. In *International Workshop on Logic Synthesis*, MCNC, Research Triangle Park, NC, May 1991.


TU2
TU1
TU4
TU3
In
K4
K2 K4
K1 K3
LM3
LM2
LM1
LA2
RA2
LA1
RA1
M3
M2
M1
A2
A1
Out

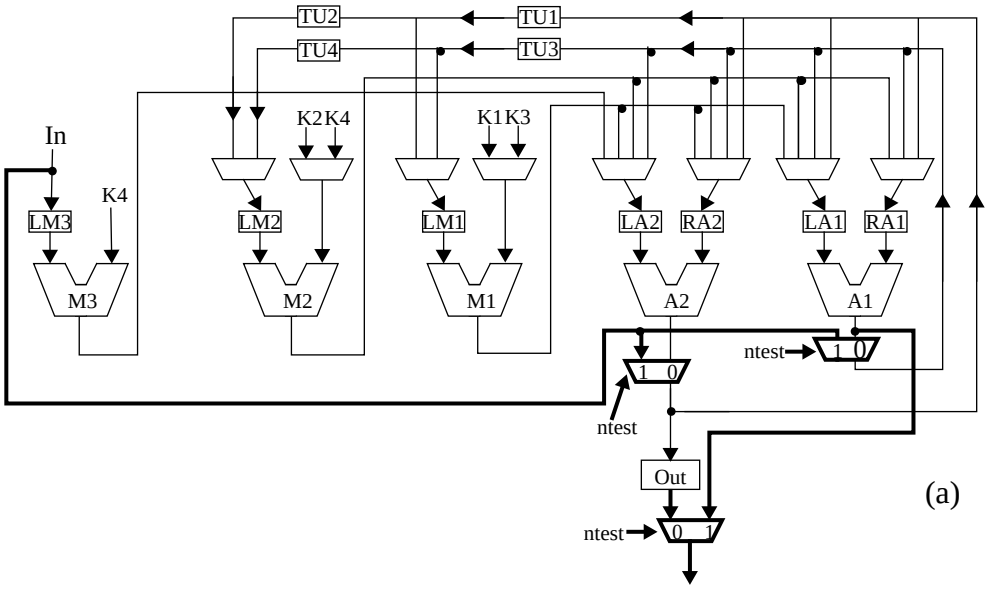
(a)


PI
LM3
LA2
RA2
LA1
RA1
OUT
TU1
TU3
LM1
TU2
TU4
LM2

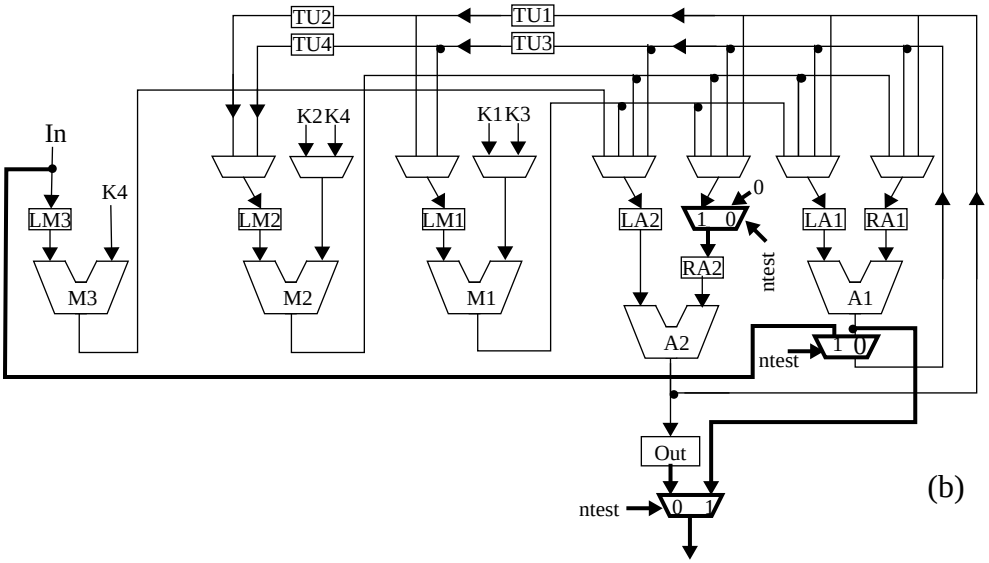
(b)


PI
1
1, 2
M3
M2
M1
A2
A1
2
Out
PO
TU1
TU2
TU3
TU4

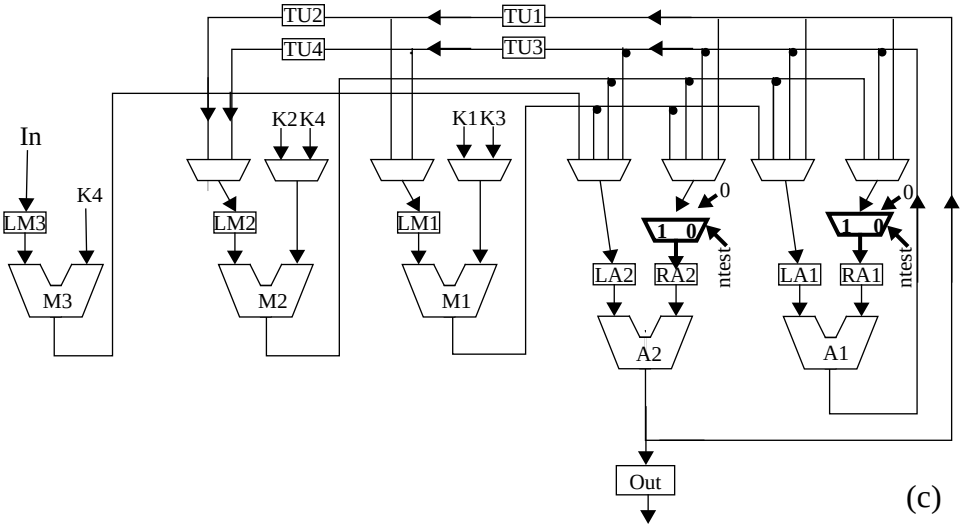
(c)


TU2
TU1
TU4
TU3
In
K2K4
K1K3
K4
LM3
LM2
LM1
LA2
RA2
LA1
RA1
M3
M2
M1
A2
A1
ntest
1 0
ntest
Out
ntest
0 1

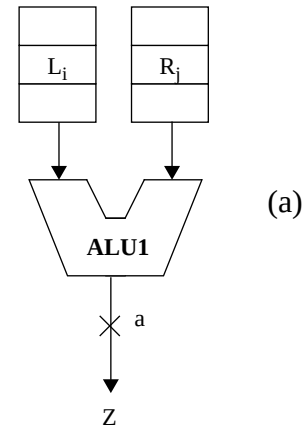
(a)


TU2
TU1
TU4
TU3
In
K2K4
K1K3
K4
LM3
LM2
LM1
LA2
0
1 0
LA1
RA1
RA2
ntest
M3
M2
M1
A1
A2
ntest
1 0
Out
ntest
0 1

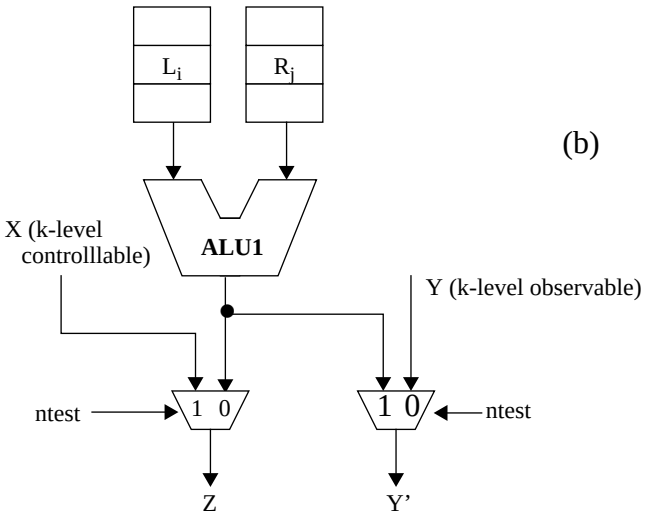
(b)


TU2
TU1
TU4
TU3
In
K2K4
K1K3
K4
LM3
LM2
LM1
0
0
1 0
1 0
LA2
RA2
ntest
LA1
RA1
ntest
M3
M2
M1
A2
A1
Out

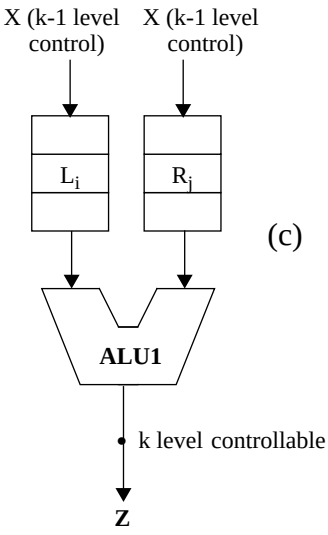
(c)


L_i
R_j
ALU1
a
Z

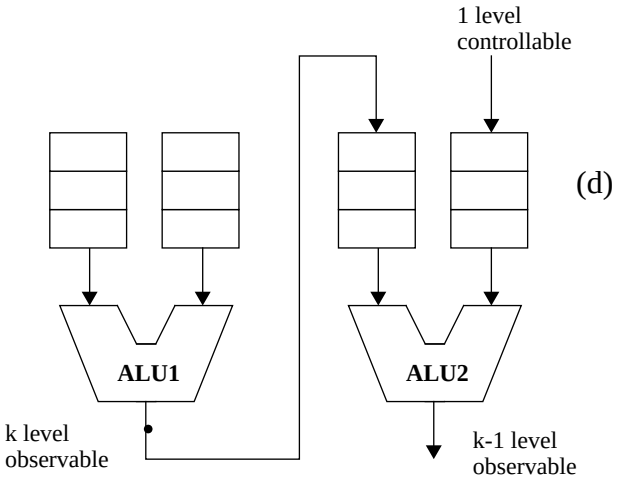
(a)


L_i
R_j
X (k-level controlllable)
ALU1
Y (k-level observable)
ntest
1 0
1 0
ntest
Z
Y'

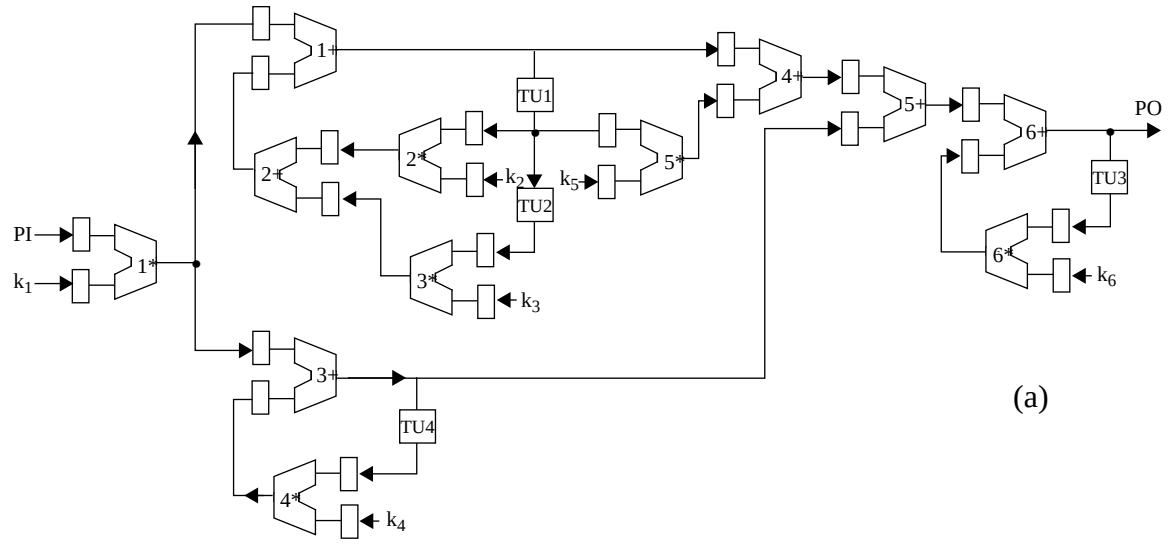
(b)


X (k-1 level control)
X (k-1 level control)
L_i
R_j
ALU1
k level controllable
Z

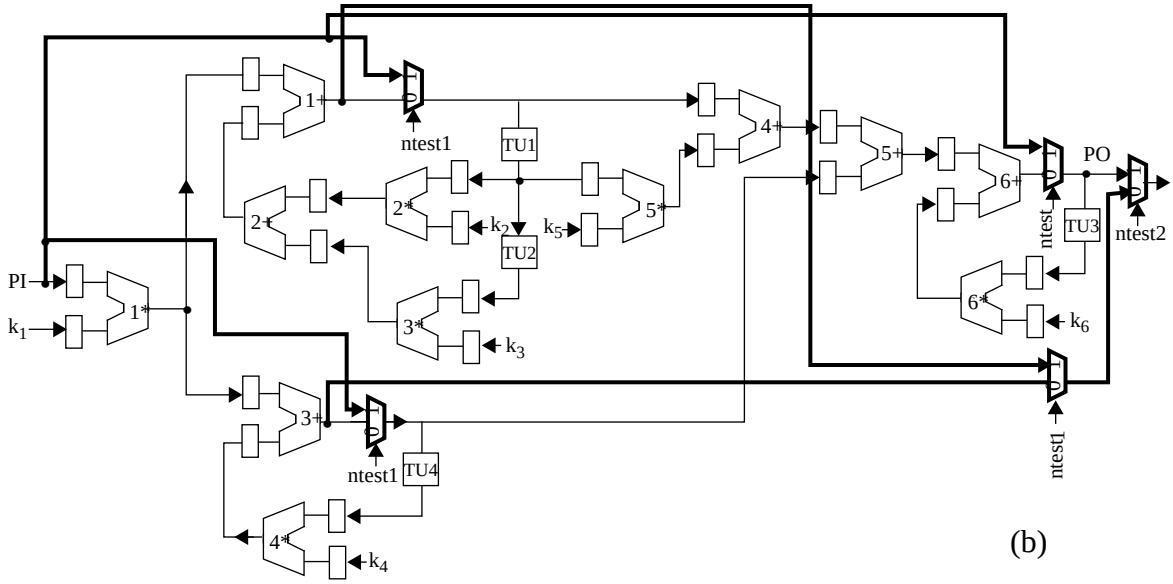
(c)


1 level controllable
ALU1
ALU2
k level observable
k-1 level observable

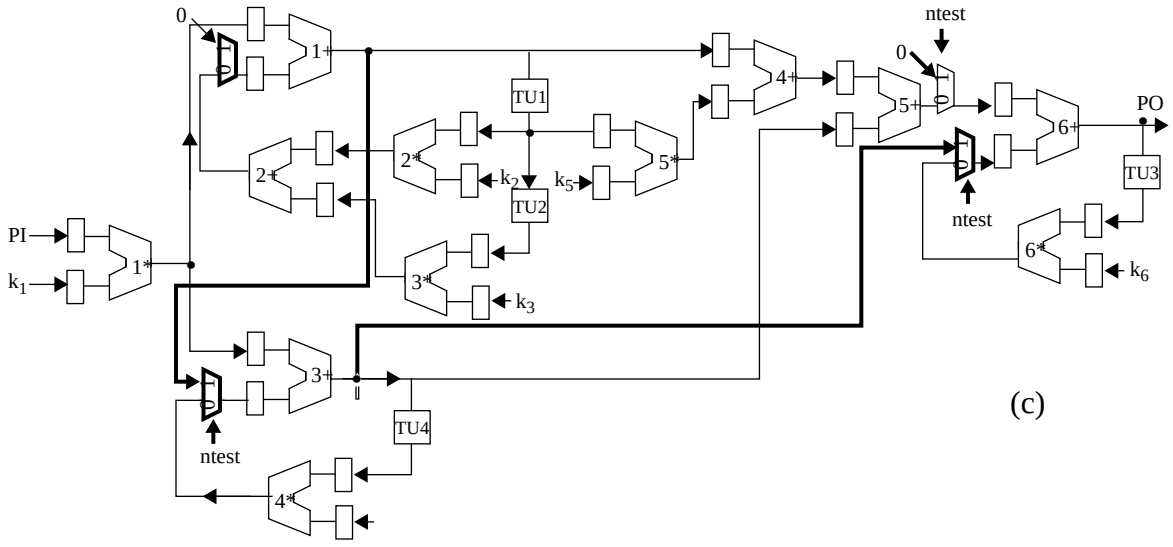
(d)


1+
2+
3+
4+
5+
6+
1*
2*
3*
4*
5*
6*
TU1
TU2
TU3
TU4
PI
k1
k2
k3
k4
k5
k6
PO

(a)


1+
2+
3+
4+
5+
6+
1*
2*
3*
4*
5*
6*
TU1
TU2
TU3
TU4
PI
k1
k2
k3
k4
k5
k6
PO
ntest1
ntest2
ntest

(b)


0
1+
2+
3+
4+
5+
6+
1*
2*
3*
4*
5*
6*
TU1
TU2
TU3
TU4
PI
k1
k2
k3
k5
k6
PO
ntest

(c)