Non-Scan Design for Testable Data Paths Using Thru Operation

Katsuyuki Takabatake[†]
Toshimitsu Masuzawa[‡]

Hokuriku Multimedia Service Promotion Headquater[†]
NTT
Kanazawa, 920 Japan
Tel:+81-762-20-9260, Fax:+81-762-60-3935
katsu@hokuriku.hkr.ntt.jp

Michiko Inoue[‡]
Hideo Fujiwara[‡]

Graduate School of Information Science[‡]
Nara Institute of Science and Technology (NAIST)
Ikoma, 630-01 Japan
Tel:+81-7437-2-5223 ,Fax:+81-7437-2-5229
{kounoe,masuzawa,fujiwara}@is.aist-nara.ac.jp

Abstract— We present a new non-scan DFT technique for register-transfer (RT) level data paths. In the technique, we add thru operations to some operational modules to make the data path easily testable. We define a testable measure, weak testability, and consider the problem to make the data path weakly testable with minimum hardware overhead. We also define a measure to estimate the test generation time. Experimental results show the effectiveness of our technique and the proposed measure.

I. Introduction

Scan-based DFT techniques[1, 2, 3, 4, 5, 6, 7, 8] make test generation easier and improve fault coverage. However, such techniques have a large area overhead, and long test application time because of shifting of the test vectors. Several scan-based techniques were proposed to improve such disadvantages. Partial scan design[3, 4] reduces the number of scan flip-flops, and consequently reduces the area overhead and the test application time. To reduce the test application time more, several techniques, such as a parity-scan design[5], parallel scan chains[6] and reconfigurable scan chains[7], are proposed. A testing method H-SCAN[8] reduces both the area overhead and the test application time by considering scan chains at RT level.

However, scan-based DFT techniques have another disadvantage that the test vectors cannot be applied *at-speed*. Maxwell et al.[9] reported that test vectors for stuck-at faults applied at speed identifies more defective chips than the same test vectors applied at lower speed. Recently, some non-scan DFT techniques were proposed. Rudnick et al.[10] and Chickermane et al.[11] presented a non-scan DFT technique for gate level sequential circuits. They introduce controllability and observability points to make the circuit easily testable. Dey and Potkonjak[12] presented a non-scan DFT technique for RT level data paths. They define a new testability measure, k-level testability, and propose a method that adds test hardware to make the data path k-level testable. Ghosh et al.[13] presented another non-scan RT-level DFT technique applicable to the RT level circuits obtained by high-level synthesis. They use the control data flow graph (CDFG) to determine the points to insert test hardware to make the RT level circuit hierarchically testable. The hierarchical testability is the measure targeting hierarchical test generation.

In this paper, we focus on the non-scan DFT techniques applicable to RT level data paths. We define a new testability called *weak testability* and present a new DFT technique which uses *thru module* to make the data path weakly testable. We assume that the controller can be modified to support the test plan and to be self-testable in non-scan fashion by combination of the techniques described in [11, 14, 15].

In general, output value of an operational module depends on all its input values, therefore, controllability of the output depends on controllabilities of all inputs, and observability of an input depends both observability of the output and controllabilities of the other inputs. Thru operation propagates one designated input (*thru input*) value to the output, that is, the output value depends only on the thru input value. In our DFT technique, thru operation is added to some operational modules to make the data path easily testable. One of remarkable advantages of our technique is *no/low* area overhead. For example, in the case of ALU providing addition and subtraction operations, such as Fig.1(a), it originally provides thru operation. In Fig.1(a), thru operation with a thru input X is realized by setting $s_0 = s_1 = c_{in} = 0$. In other case, thru operation can be realized with small overhead. In Fig.1(b), thru operation with thru input X is realized by adding bit-width AND gates and set $s_0 = 0$.

The weakly testable data path guarantees that, for each hardware element (i.e., an operational module, a register, etc.), there exist a collection of paths to support justification/propagataion of its output values. To controll/observe an output of a hardware element *fully* (that is, any value on the output can be jutified from some primary inputs and propagated to some primary output), it may need to contorol independently two or more inputs of hardware elements which appears on justification/propagation paths. However, the definition of our weak testability does not consider such independency. In this sense, we use a term "weak". Though our definition is weak compared with full testability, the experimental result shows the effectness of weak testability.

We also define a new testability measure, *weak testability cost*, to estimate the test genaration time. This cost consists of two costs; *weak control cost* and *weak observation cost*. The weak control(resp. observation) cost of

ASP-DAC '97
0-89791-851-7/$5.00

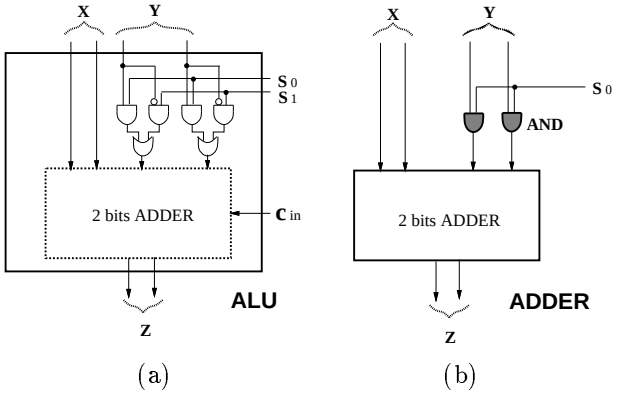

Fig. 1. An exapmle of thru module: (a)realization of thru on ALU, (b)realization on Adder.

a hardware element is measure for estimating the number of clock cycles necessary to justify(resp. propagate) some value (not any value) from(resp. to) some primary inputs(resp. outputs). From the experimental results, we confirm the correlation between the weak testability cost and the test generation time for some data paths.

The rest of the paper is organized as follows. In section II, some basic definitions and a defintion of a weak testable data path are given. In section III, we consider the problem to make a given data path weakly testable with minimum hardware overhead, and show the NP-hardness of this problem. In section IV, we mention about the weak testability cost. Experimental results are shown in section sec:exp. Conclusions are given in section VI.

II. Weak testability

A data path consists of *hardware elements* and *lines*, where a hardware element is a primary input, a primary output, a register, a multiplexor, or an operational module, and a line connects two hardware elements with some bit width. Let M be an operational module and $\mathcal{F}_M$ be a set of operation which M provides. An operational module M is *thru module* if the following holds.

$$\exists f_{th} \in \mathcal{F}_M, \exists i, f_{th}(X_1, \cdots, X_i, \cdots, X_n) = X_i.$$

We call the above operation *thru operation on* X_i and the input X_i a *thru input*. If an input X is a thru input, X is denoted by $\hat{X}$. Let $\mathcal{M}$ denote a set of operational modules, and $\mathcal{IN}_M$ denote a set of inputs of an operational module M. Let H_1 and H_2 be hardware elements, and X be an input of a hardware element. Let $H_1 \rightarrow X$ (resp. $H_1 \rightarrow H_2$) means that there is a line connecting the output of H_1 and X (resp. some input of H_2).

Now we define weak controllability/observability of a hardware element. Intuitively, weak controllability (resp. observability) of a hardware element H means that *some* value (not necessarily *any*) on the output of H can be justified (resp. propagated) from primary inputs (resp. outputs). Weak controllability is defined recursively from the primary inputs; a hardware element is weakly controllable, if all its inputs or at least one of its thru inputs are weakly controllable.

Definition 1 *weak controllability*
A set of weakly controllable hardware elements is the minimum set $\mathcal{H}_{wc}$ satisfying the following conditions.

1. For any primary input I, $I \in \mathcal{H}_{wc}$.
2. For any register or multiplexor H, $\exists H_{wc} \in \mathcal{H}_{wc}[H_{wc} \rightarrow H] \Rightarrow H \in \mathcal{H}_{wc}$.
3. For any operational module M,

$$\begin{array}{l}\forall X \in \mathcal{IN}_M\ [\exists H_{wc} \in \mathcal{H}_{wc}\ [H_{wc} \rightarrow X]] \vee \\ \exists \hat{X} \in \mathcal{IN}_M\ [\exists H_{wc} \in \mathcal{H}_{wc}\ [H_{wc} \rightarrow \hat{X}]] \\ \Rightarrow M \in \mathcal{H}_{wc}\,!\%\end{array}$$

It can be considered that some value of some input of a hardware element is propagated (in a weak sense) to its output either if the input is thru input or if all other inputs are weakly controllable. Weak observability is defined recursively from the primary outputs; a hardware element is weakly observable, if its output values can be propagated (in a weak sense) to an output of some weakly observable hardware element.

Definition 2 *weak observability*
A set of weakly observable hardware elements is the minimum set $\mathcal{H}_{wo}$ satisfying the following conditions.

1. For any primary output O, $O \in \mathcal{H}_{wo}$.
2. For any hardware element H,

$$\begin{array}{l}\exists H_{wo} \in \mathcal{H}_{wo} - \mathcal{M}\ [H \rightarrow H_{wo}] \vee \\ \exists M_{wo} \in \mathcal{H}_{wo} \cap \mathcal{M}[\\ \quad \exists X \in \mathcal{IN}_{M_{wo}}[H \rightarrow X \wedge \forall X' \in \mathcal{IN}_{M_{wo}} - \{X\} \\ \qquad [H \rightarrow X' \vee \exists H_{wc} \in \mathcal{H}_{wc}\ [H_{wc} \rightarrow X']]] \vee \\ \quad \exists \hat{X} \in \mathcal{IN}_{M_{wo}}\ [H \rightarrow \hat{X}]] \\ \Rightarrow H \in \mathcal{H}_{wo}\,!\%\end{array}$$

Definition 3 *weak testability*
A data path DP is weakly testable iff all registers in DP are weakly controllable and weakly observable.

It is obvious from the definition that ,if all registers in a data path are weakly controllable, all registers are also weakly observable, that is, the data path is weakly testable.

For example, in a data path of the 4th order IIR cascade filter in Fig.2, only a primary input In, registers LM3, LA2 and an operational module M3 are weakly controllable. An operational module A2 is not weakly controllable, since a register RA2 is not weakly controllable. However, a register LA2 is weakly controllable, therefore, A2 can become weakly controllable by adding thru operation on the input X_4 to A2. In this case, all registers consequently become weakly controllable.

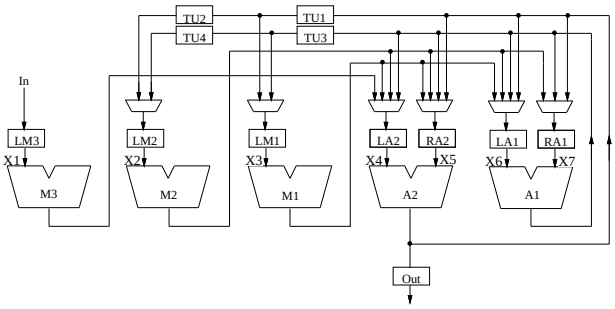

Fig. 2. RT-level data path of the 4th order IIR cascade filter.

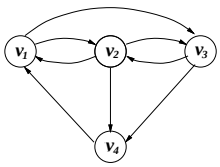

Fig. 3. An example of directed graph G'.

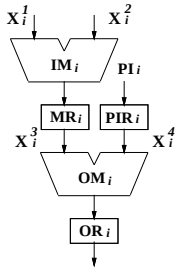

Fig. 4. Hardware structure corresponding to a vertex v_i of directed graph G'.

III. Minimum thru input set problem

A. NP-hardness

For a data path DP and a set IN of some inputs of operational modules in DP, let DP' be a data path obtained from DP by making all inputs in IN thru inputs. If DP' is weakly testable, we say that IN is a *thru input set*, or *TIS*, of DP. We consider a problem to find a minimum size TIS (MTIS) of a given data path.

Definition 4 *Minimum thru input set problem (MTISP)*
Input: *A data path DP.*
Output: *An MTIS of DP.*

We first show NP-hardness of the MTISP by reduction from the following problem 2-MFVSP which is known to be NP-hard [16]. For a directed graph $G = (V, E)$, a vertex set $F(\subseteq V)$ is called a *feedback vertex set*, or *FVS*, if F contains at least one vertex from each loop.

Definition 5 *2-Minimum feedback vertex set problem (2-MFVSP)*
Input: *A directed graph $G = (V, E)$ whose maximum in-degree is at most 2.*
Output: *A minimum size FVS (MFVS) of G.*

Transformation to a data path: We transform a directed graph $G = (V, E)$ whose maximum in-degree is at most 2 to a data path DP. Using MFVS-preserving transformations described in [17], G can be transformed to a directed graph $G' = (V', E')$ where in-degree of each vertex in V' is exactly 2, and a minimum FVS of G can be computed easily from G'. For convenience, we assume that $V' = \{v_1, v_2, \cdots, v_n\}$.

We transform each vertex $v_i \in V'$ to the following hardware elements and lines (Fig.4).

- hardware elements: 2-input operational modules $IM_i(\mathcal{IN}_{IM_i} = \{X_i^1, X_i^2\})$, $OM_i(\mathcal{IN}_{OM_i} = \{X_i^3, X_i^4\})$, registers PIR_i, MR_i, OR_i, and a primary input PI_i.
- lines : $IM_i \rightarrow MR_i$,$MR_i \rightarrow X_i^3$,$PI_i \rightarrow PIR_i$,$PIR_i \rightarrow X_i^4$,$OM_i \rightarrow OR_i$.

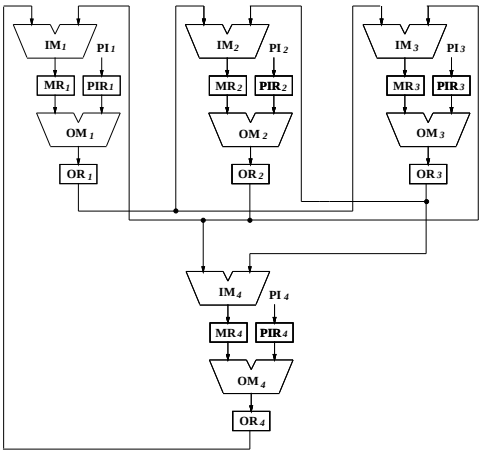

Fig. 5. Data path corresponding to G'.

For each v_i, Two incoming edges $(v_j, v_i), (v_k, v_i) \in E'$ are transformed into lines $OR_j \rightarrow X_i^1$, $OR_k \rightarrow X_i^2$. For example, a directed graph in Fig.3 is transformed into a data path in Fig.5.

The above transformation from G to DP can be executed in polynomial time for the size of G. In the following, we show how to compute an MFVS of G from an MTIS of DP.

Lemma 1 *If T is a TIS of DP, $F = \{v_i | \exists k[X_i^k \in T]\}$ is an FVS of G'.*

Proof. Assume that F is not an FVS of G', that is, there is a loop $C = (v_{a_1}, v_{a_2}, \cdots, v_{a_1})$ in G' such that F does not contain any vertex in C. In this case, DP has a loop $C' = (OR_{a_1} \rightarrow IM_{a_2} \rightarrow MR_{a_2} \rightarrow OM_{a_2} \rightarrow OR_{a_2} \rightarrow \cdots \rightarrow IM_{a_1} \rightarrow MR_{a_1} \rightarrow OM_{a_1} \rightarrow OR_{a_1})$, and any $X_{a_i}^1$, $X_{a_i}^2$, $X_{a_i}^3$, $X_{a_i}^4$ is not in T (otherwise $v_{a_i} \in F$). However, for DP to be weakly testable, some operational module in C' must be a thru module. That is, T is not a TIS of DP. A contradiction occurs. ■

Lemma 2 *If F is an FVS of G', $T = \{X_i^4 | v_i \in F\}$ is a TIS of DP.*

Proof. Assume that T is not a TIS of DP. Let DP' is a data path obtained from DP by making all inputs in T thru inputs. Since T is not a TIS of DP, there exist a register R in DP' such that R is not weakly controllable. In DP', any PIR_i is weakly controllable, therefore, some OR_i or MR_j is not weakly controllable. If OR_i is not weakly controllable, then an input X_i^4 is not a thru input and MR_i is not weakly controllable. If MR_j is not weakly controllable, there exists some OR_k such that OR_k is not weakly controllable and $OR_k \rightarrow IM_j$. Since the number of hardware elements in DP' is finite, there exists a loop $OR_{a_1} \rightarrow IM_{a_2} \rightarrow MR_{a_2} \rightarrow OM_{a_2} \rightarrow OR_{a_2} \rightarrow IM_{a_3} \rightarrow \cdots \rightarrow OR_{a_1}$ such that any $X_{a_i}^4$ is not a thru input, that is, $v_{a_i} \notin F$. However, $(v_{a_1}, v_{a_2}, v_{a_3}, \cdots, v_{a_1})$ is a loop in G', that is, F is not an FVS of G'. A contradiction occurs. ■

Lemma 3 *For each $i(1 \leq i \leq n)$, an MTIS of DP contains at most one of X_i^1, X_i^2, X_i^3 and X_i^4.*

Proof. Assume that two inputs X_i^p, X_i^q $(p \neq q)$ are in some MTIS T of DP. From Lemma1 and Lemma2, $F' = \{v_i | \exists k[X_i^k \in T]\}$ is an FVS of G', and $T' = \{X_i^4 | v_i \in F'\} = \{X_i^4 | \exists k[X_i^k \in T]\}$ is a TIS of DP. Since $X_i^p \neq X_i^q$, $|T'| < |T|$ holds, that is, T is not an MTIS of DP. A contradiction occurs. ■

Lemma 4 *If T is an MTIS of DP, $F = \{v_i | \exists k[X_i^k \in T]\}$ is an MFVS of G'.*

Proof. From Lemma1, F is an FVS of G', and from Lemma3, $|T| = |F|$ holds. Assume that F is not an MFVS of G', that is, there exist an FVS F' of G' such that $|F'| < |F|$. From Lemma 2, $T' = \{X_i^4 | v_i \in F'\}$ is a TIS of DP, and $|F'| = |T|$. Therefore, $|T'| < |T|$ holds. This contradicts that T is an MTIS of DP. ■

From Lemma4, we can easily compute an MFVS of G from an MTIS of DP. Therefore, the following theorem holds.

Theorem 5 *The problem MTISP is NP-hard.*

B. Heuristics

We present a heuristic algorithm for the MTISP (a pseudo code in Fig.6). In this algorithm, weakly controllable hardware elements and module inputs are marked from primary inputs according to the definition. If there remain some unmarked elements, we select one effective(see the pseudo code) module input, and set it a thru input. The above process is repeated until all elements are marked.

This algorithm does not guarantee to find MTIS, however, for two benchmark data paths used in our experiments, it finds MTISs.

IV. Weak testability cost

We can consider that test generation time for data paths depends on the number of clock cycles necessary to justify/propagate values of registers from/to primary inputs/outputs. Here, we introduce measure to estimate this number of clock cycles.

```
find_thru_inputs()
  M = {operational modules};
  N = {hardware elements} - M;
  I = {module inputs};
  /* initially
       all elements in M ∪ N ∪ I are unmarked,
       thru_candidate and thru_inputs are empty */
  set all primary inputs ready;
  while ∃ an unmarked element {
    while ∃ a ready element {
      select a ready element e; set e marked;
      if e ∈ M ∪ N
        set any unmarked element in S ready
        where S = {h ∈ N ∪ I | e → h};
      else /* e is an input of some module M */
        if all inputs in IN_M are marked
        or e is thru input
          delete M from thru_candidate;
          if M is unmarked
            set M ready;
        else add M to thru_candidate;
    }
    if thru_candidate ≠ ∅
      select M from thru_candidate
      with max |{h ∈ N ∪ I | h is unmarked, M → h}|;
      delete M from thru_candidate;
      select one marked input X in IN_M
      add X to thru_inputs;
  }
  return thru_inputs
```

Fig. 6. TIS finding algorithm

A *weak control cost* of a hardware element H, denoted $wcc(H)$, is a measure to estimate the number of clock cycles necessary to justify some output value of H from primary inputs.

Definition 6 For each hardware element H, a weak control cost of H, denoted by $wcc(H)$, is defined as follows.

1. For a primary input I,
   $wcc(I) = 0$.
2. For a register R and a hardware element H such that $H \rightarrow R$,
   $wcc(R) = wcc(H) + 1$.
3. For a multiplexor S,
   $wcc(S) = \min\{wcc(H) \mid H \rightarrow S\}$
4. For an operational module M,
   (a) if M is not a thru module,
   $wcc(M) = \max\{wcc(H) \mid X \in \mathcal{IN}_M\ [H \rightarrow X]\}$,
   (b) otherwise (M is a thru module),
   $wcc(M) = \min\{wcc(H) \mid \hat{X} \in \mathcal{IN}_M\ [H \rightarrow \hat{X}]\}$.

A *weak observation cost* of a hardware element H, denoted $woc(H)$, is a measure to estimate the number of

clock cycles necessary to propagate some output value of H to an primary output. To propagate some value from an input to an output of some hardware element, it may need to justify some values on the other inputs. To propagate some value of H to some primary output via some propagation path P, it may need for some hardware elements on P to justify some values on their inputs. In some case, the weak control cost of some input X of some hardware element H' on P may be larger than the number of clock cycles to propagate a value from H to H' via P. The weak observation cost considers such *in-advance* weak control cost. In the following definition, $\mathcal{L}(X)$ represents a set of pairs of the length a propagation path P and the in-advance weak control cost of P for all propagation paths of X.

Definition 7 For each hardware element H, a weak observation cost of H, denoted by $woc(H)$, is defined as follows.

1. For a primary output U and its input X_U,
$$\begin{cases} woc(U) = 0 \\ \mathcal{L}(X_U) = \{(0,0)\} \end{cases}$$

2. For a register R and its input X_R,
$$\begin{cases} woc(R) = \min\{(d+p) \mid (d,p) \in \bigcup_{R \to X} \mathcal{L}(X)\} \\ \mathcal{L}(X_R) = \{(d+1, \max(p-1,0)) \\ \qquad\qquad \mid (d,p) \in \bigcup_{R \to X} \mathcal{L}(X)\} \end{cases}$$

3. For a multiplexor S and its input X_S,
$$\begin{cases} woc(S) = \min\{(d+p) \mid (d,p) \in \bigcup_{S \to X} \mathcal{L}(X)\} \\ \mathcal{L}(X_S) = \{(d,p) \mid (d,p) \in \bigcup_{S \to X} \mathcal{L}(X)\} \end{cases}$$

4. For an operational module M and its $X_M \in \mathcal{IN}_M$,
$$woc(M) = \min\{(d+p) \mid (d,p) \in \bigcup_{M \to X} \mathcal{L}(X)\}$$

   (a) if X_M is not a thru input
$$\mathcal{L}(X_M) = \{(d, \max(mcc(X_M), p)) \mid (d,p) \in \bigcup_{M \to X} \mathcal{L}(X)\}$$
   where $mcc(X_M) = \max\{wcc(X) \mid X \in \mathcal{IN}_M - \{X_M\}\}$.

   (b) otherwise (X_M is a thru input) $\mathcal{L}(X_M) = \{(d,p) \mid (d,p) \in \bigcup_{M \to X} \mathcal{L}(X)\}$

For a data path DP, we define a *weak testability cost* as sum of weak controllability costs and weak observability costs of all registers.

Definition 8 For a data path DP, let $\mathcal{R}$ be a set of registers in DP. A weak testability cost of DP, denoted by $wtc(DP)$, is defined as follows.

$$wtc(DP) = \sum_{R \in \mathcal{R}} (wcc(R) + woc(R))$$

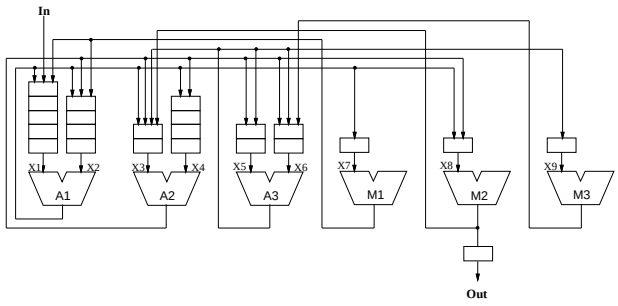

Fig. 7. RT-level data path of the 5th order wave digital elliptocal filter

V. Experimental Results

To evaluate our DFT technique and the weak testability cost, we applied our technique and other techniques on RT-level data paths, 4th order IIR cascade filter (4th IIR) in Fig.2 and 5th order digital elliptical filter (5th EWF) in Fig.7. Table I shows the hardware resources of each data path, where *bists* denotes the word size, *#add* denotes the number of adders, *#mult* denotes the number of multipliers, and *#reg* denotes the number of registers. Our experiment used a graphical functional design system Bchart (Matsushita Electric Ind. Co.) which generates a VHDL description from a functional diagram, a logic synthesis tool AutoLogic (Mentor Graphics Co.), and an ATPG tool TestGen (Sunrise Test System, Inc.) on a SUN SPARCstation10.

Tables II and III show the results. In these tables, the columns show a type of data path (*type*), hardware overhead added to the original data-path to make it testable (*overhead*), the number of gates after a logic synthesis(*#gates*), the weakly testable cost(*wtc*), the number of all stuck-at faults (*#faults*), test efficiency (*test eff.*), the test generation time(*CPU[sec]*) and the test application time (the number of test vectors)(*apll.[cycle]*). We made experiment on several design types. The type *orig* denotes an original data path (data path applied no DFT technique), LR is a data path obtained by the partial scan method presented in [4], and DP is a 0-level testable data path obtained by the non-scan DFT technique presented in [12] where 0-level data path has the highest testability in terms of the testability measure "k-testability" proposed in [12]. Data paths added some through inputs to be weakly testable are T1,$\cdots$,T6 in Table II and T1,$\cdots$,T9 in Table III. In the column *overhead*, TI de-

TABLE I
Hardware resources of each data path.

design	*bits*	*#add*	*#mult*	*#reg*
4th IIR	10	2	3	12
5th EWF	10	3	3	23

TABLE II
EXPERIMENTAL RESULT1: 4TH IIR

type	*overhead* (thru inputs)	*#gates*	*wtc*	*#faults*	*test eff.* [%]	*CPU* [sec]	*appl.* [cycle]
orig	–	5827	–	12536	13.97	>6hr	–
LR	3 scan registers	5947	–	12656	99.83	234	4950
DP	3 multiplexors	5947	–	12776	99.99	72	464
T1	1 TI(X4)	5837	72	12596	99.98	188	426
T2	2 TIs(X4,X5)	5847	70	12656	100.00	87	492
T3	2 TIs(X4,X6)	5847	69	12656	100.00	69	570
T4	3 TIs(X4,X5,X7)	5857	67	12716	100.00	67	602
T5	4 TIs(X4,X5,X6,X7)	5867	64	12776	100.00	64	506
T6	7 TIs(all inputs)	5900	64	12956	100.00	66	704

TABLE III
EXPERIMENTAL RESULT2: 5TH EWF

type	*overhead* (thru inputs)	*#gates*	*wtc*	*#faults*	*test eff.* [%]	*CPU* [sec]	*appl.* [cycle]
orig	–	6576	–	15652	19.52	>8hr	–
LR	15 scan registers	7176	–	16252	99.46	497	19583
DP	6 multiplexors	6816	–	16132	98.12	1428	313
T1	1 TI(X1)	6586	128	15712	97.47	2116	390
T2	2 TIs(X1,X4)	6596	120	15772	98.00	1777	1054
T3	3 TIs(X1,X3,X4)	6606	116	15832	98.00	1567	905
T4	4 TIs(X1–3,X5)	6616	114	15892	98.11	1519	836
T5	4 TIs(X1,X3–5)	6616	110	15892	98.08	1512	832
T6	4 TIs(X1,X3,X4,X6)	6616	108	15892	98.16	1417	825
T7	5 TIs(X1–3,X5,X6)	6626	106	15952	98.17	1404	825
T8	6 TIs(X1–6)	6636	98	16012	98.18	1347	972
T9	9 TIs(all inputs)	6669	98	16192	99.55	1005	892

notes through input. Especially, T1 in each Table II, III is obtained by our heuristic algorithm to find a thru input set, and an minimum thru input set is found as a result.

Table II shows that all DFT technique obtain very high test efficiency (more than 99%) for the original data path of test efficiency 13.97%. In these highly testable design types, T1 has the lowest hardware overhead (see the columns *#gates* and *overhead*) and the smallest test application time. Table III shows that our DFT method (T1) can increase test efficiency to 97.47% from original 19.52%. As compared with LR, DP, this T1 has lower test efficiency, but the type T9 shows that we can obtain higher test efficiency with lower hardware overhead than LR, DP. We can also see the effectness of the weakly testable cost *wtc*. Tables II and III show the correlation between the weak testability cost and the test generation time, which implies that the weak testability cost can be used to estimate test generation time.

VI. CONCLUSION

In this paper, we introduced a non-scan DFT technique using thru operations applicable to RT-level data paths. Non-scan DFT techniques have advantages of short test application time and possibility of at-speed test. We proposed a testability called weak testability, and considered the problem to make a given data path weakly testable. We showed this problem is NP-hard, and presented a heuristic algorithm. We also proposed a measure, weak testability cost, to estimate test generation time. Experimental results show that our technique obtains high test efficiency with low hardware overhead and that the weak testability cost has correlation with the test generation time. We considered DFT using thru modules *after* RT-level design was finished. One of our future works is to consider SFT (synthesis for testability) using thru modules *during* high-level synthesis.

REFERENCES

[1] E. B. Eichelberger and T. W. Williams: "A logic design structure for LSI testability", Journal of Design Automation and Fault-Tolerant Computing, **2**, 2, pp. 165–178 (1978).

[2] M. Williams and J.B.Angell: "Enhancing testability of large scale integrated circuits via test points and additional logic", IEEE Transaction on Computers, **C-22**, 1, pp. 46–60 (1973).

[3] S. T. Chakradhar, A. Balakrishnan and V. D. Agrawal: "An exact algorithm for selecting partial scan flip-flops", Journal of Electronic Testing: Theory and Applications, **7**, pp. 83–93 (1995).

[4] D. H. Lee and S. M. Reddy: "On determining scan flip-flops in partial-scan design approach", Proceedings of the International conference on Computer-Aided Design, pp. 322–325 (1990).

[5] H. Fujiwara and A. Yamamoto: "Parity-scan design to reduce the cost of test application", Proceedings of the International Test Conference, pp. 283–292 (1992).

[6] B. Vinnakota and N. K. Jha: "Synthsis of sequential circuits for parallel scan", Proceedings of the European Conference on Design Automation, pp. 366–370 (1992).

[7] S. Narayanan and M. A. Breuer: "Reconfigutable scan chains: A novel approach to reduce atest application time", Proceedings of the International Conference on Computer-Aided Design, pp. 710–715 (1993).

[8] S. Bhattacharya and S. Dey: "H-SCAN:a high level alternative to full-scan testing with reduced area and test application overheads", Proceedings of the VLSI Test Symposium, pp. 74–80 (1996).

[9] 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%?,", Proceedings of the International Test Conference, pp. 358–364 (1991).

[10] E. M. Rudnick, V. Chickermane and J. H. Patel: "Probe point insertion for at-speed test", Proceedings of the VLSI Test Symposium, pp. 223–228 (1992).

[11] V. Chickermaned, E. M. Rudnick, P. Banerjee and J. H. Patel: "Non-scan design-for-testability techniques for sequential circuits", Proceedings of the Design Automation Conference, pp. 236–241 (1993).

[12] S. Dey and M. Potkonjak: "Non-scan design-for-testability of RT-level data paths", Proceedings of the International Conference on Computer-Aided Design, pp. 640–645 (1994).

[13] I. Ghosh, A.Raghunathan and N. K. Jha: "Design for hierachical testability of RTL circuits obtained by behavioral synthesis", Proceedings of the Intenational Confernce on Computer Design, pp. 173–179 (1995).

[14] S. Hellebrand and H. Wumderlich: "Synthesis of self-testable controllers", Proceedings of the European Conference on Design Automation, pp. 580–585 (1994).

[15] M. P. S. Dey, V. Gangaram: "A controller-baced design-for-testability technique for controller-data path circuits", Proceedings of the International Conference on Computer-Aided Design, pp. 534–540 (1995).

[16] M. R. Garey and D. S. Johnson: "Computers and Intractability — A Guide to the Theory of NP-Completeness —", Freeman (1991).

[17] D. H. Younger: "Minimum feedback vertex sets for directed graphs", IEEE Trans. on Circuit Theory, **10**, pp. 238–245 (1963).