Asynchronous Scan-Latch controller for Low Area Overhead DFT

Masayuki Tsukisaka, Masashi Imai, and Takashi Nanya
Research Center for Advanced Science and Technology, The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8904, JAPAN
{tsukky, miyabi, nanya}@hal.rcast.u-tokyo.ac.jp

Abstract

This paper introduces a new scan control technique to realize low area overhead of scan-latches. Single transparent-latch is popularly used for register of high-throughput datapaths. For the scan-test of those kind of circuits, each transparent-latch is replaced with scan-latch. Conventional scan-latch cells controlled by synchronous signals consist of L1 latch and additional L2 latch, both of which function as master latch and slave latch respectively in scan mode. Apparently, additional L2 latch may result in area overhead. In order to avoid the area impact of such an additional L2 latch, we propose new timing methodology employing asynchronous control technique **asP*** *protocol, and introduce asynchronous controlled scan-paths whose scan-latch employs only L1 latch. We evaluate the operation speed with* **HSPICE** *simulations and see they are practical. We also suggest DFT structure with our suggested asynchronous scan-paths, which is suitable for conventional synchronous test systems.*

1. Introduction

With the advancement of sub-micron technologies and increase of circuits complexities, the significance of design for testability(DFT) is going more important in spite of its several drawbacks, area overhead and power dissipation in scan mode.

Normally, scan registers are controlled by synchronous signals. So each bit of scan cells requires at least two latches, L1, L2, which function as master-slave latches to shift test vectors safely(Fig.1(a)). When circuits under test(CUT) employ master-slave flipflops as their registers, both L1 and L2 in scan cell can be used in normal mode. The problem is that CUT employ single transparent latches as their registers. One of L1, L2 in the scan cell is redundant in normal mode, and the redundant latch apparently cause area overhead.

Therefore, many test engineers make an effort to design smaller scan-latches and smaller scan path structure. Special scan-latches are employed to achieve lower area scan path at the expense of circuits' stableness[4]. The special scan-latch consists of normal sized L1 and small sized L2 which has no memory element. Actually, the area overhead of this special scan-latch is low, but timing constraints are severe. Another technique is *LSSD Single L2* Latch Design* suggested in [2], but this cannot be an essential solution. In this technique, both scanned and non-scanned registers are used to realize master-slave latches in scan mode. Scanned latches and non-scanned ones are mutually connected in series to shift test vectors. Actually, *LSSD Single L2* Latch Design* has no redundant latches in both scan and normal mode, but requires more complex wire-routing to form a scan path, which evidently causes area overheads [6].

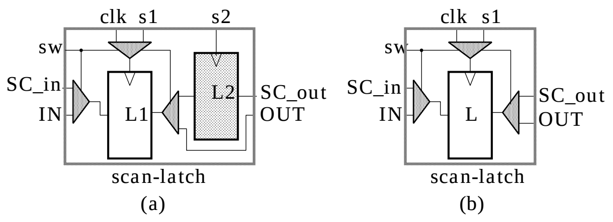

Figure 1. Cell of Scan Latch

This paper focuses on timing methodology to tackle such scan latch area problems. In normal synchronous shift registers, all bits of data shift at the same time, so each bit of the registers requires at least two latches for safe operation. However, if each bit of data can shift at the different time like *domino*, employing some smart signal controllers, single latch is enough to realize each bit of shift register(Fig.1(b)).

In the next section, our timing methodology for shift register control is shown.

2. Multi-Clocked Shift-Register

To introduce our timing methodology for low area overhead, we briefly study the behavior of synchronous shift register.

Fig.2(a) shows the conventional synchronous shift registers and their control signal. Each stage of L represents single transparent latch. In each clock cycle, data in the shift register can shift forward by a bit.

Generally, N-stages of synchronous shift registers can shift at most $N/2$ bit of data, because every time control signal clocked, all stages of the latches behave as master or slave latches.

Fig.2(b) shows the shift register with our suggested timing methodology. This shift register is controlled by n-pulse signals. In each clock cycle, data in the shift register can shift forward by a bit. Therefore, N-stages of such multi-clocked shift register can shift at most (N-1) bit of data.

This multi-clocked shift register can be applied for scan register, and can realize low area overhead scan latches. The drawback of this multi-clocked scan register is operation speed. Especially, with the increase of N, the shift speed of the scan register is degrading.

For the practical operation speed, the generator of multi-clocked signals has to operate fast. We employ high speed asynchronous technique, **asP***[5] for the multi-clocked generation. Next section, **asP*** is briefly studied.

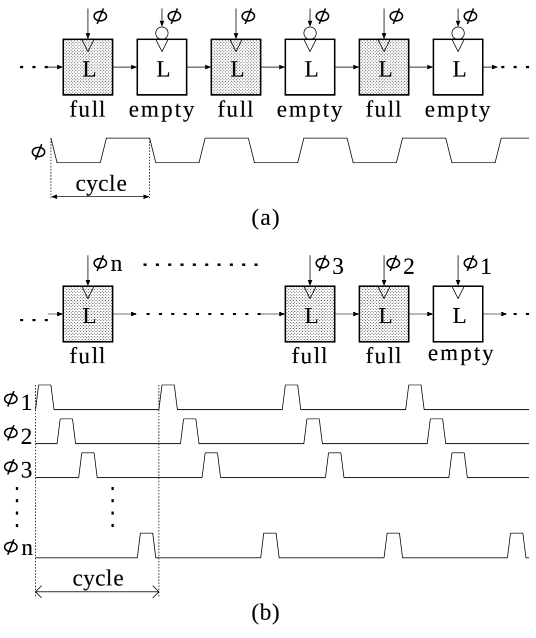

Figure 2. Shift Registers and their Control Signals

3. Asynchronous FIFO based on asP*

asP* is the abbreviation of **A**synchronous **S**ymmetric **P**ulse **P**ersistent **P**rotocol. **asP*** was proposed to realize high throughput asynchronous FIFO control, introducing *pulse like control signal*, accepting *simultaneous events*. Essentially, the behavior of **asP*** is similar to well-known four-phase[3], or return to zero asynchronous protocol, but timing constraints are harder.

Fig.3(a) shows an example of asynchronous FIFO employing **asP*** protocol. Each of the latches, $L_i (1 \leq i \leq 5)$ is controlled by **asP*** based asynchronous controller, c_i. c_i has communication with $c_{(i-1)}$ at the preceding stage accordingly with **asP*** protocol, and produce a pulse signal en_i for latch controlling. In **asP***, two states are defined at every FIFO stage, *full* and *empty*. *full* at the ith stage implies that the memory of L_i is written. *empty* implies that the memory of L_i is re-writable. c_i indicates the state of the ith stage.

When c$_{(i-1)}$ and c_i indicate *full* and *empty* respectively, c_i produces a pulse signal en_i to write an image of L_{i-1} to L_i and then the state of the $(i-1)$th is set to *empty* and the ith's is *full*. In **asP*** protocol, the generation of en$_i$ depend only on the states of the $(i-1)$th and the ith. The state of the $(i+1)$th has no effect on the relation between c$_{(i-1)}$ and c$_i$.

asP* can be applied not only linear FIFO but also join and fork structures. In any case, the condition that given stages produce enabling signals is only the case that all of the current stages are *empty* and all of their preceding stages are *full*(fig.3(b)).

Fig.3(c) shows the signal interactions between the $(i-1)$th and the ith stages. Signal $s_{(i-1)}$ and s_i is used as state indication signal in the $(i-1)$th and the ith stages respectively, HIGH level indicates *empty*, and LOW indicates *full*. Initially, both $s_{(i-1)}$ and s_i are HIGH level, which indicates that they are *empty*. When $s_{(i-1)}$ falls down to LOW level, which implies that the $(i-1)$th stage changes to *full*, following three events are issued simultaneously.

- *event1*: s_{i-1} raises to HIGH level.
- *event2*: en_i behaves as a pulse signal.
- *event3*: s_i falls down to LOW level.

As the result of these events, the data shift from L_{i-1} to L_i, and the $(i-1)$th is *empty* and the ith is *full*. For correct operations, designers have to adjust the delays of those events and the period of a pulse to satisfy SetupTime and HoldTime of the latches at every stage of FIFO. Therefore, the designers have to pay attention to the following timing constraints.

- In *event1*, s_{i-1} has to raise after pulse signal finished.

- In *event2*, a pulse signal has to begin after the data of stage $(i-1)$ arrives at stage i.
- In *event2*, a pulse signal has to finish after the latch of stage i is written.
- In *event3*, s_i has to fall after the pulse signal finished.

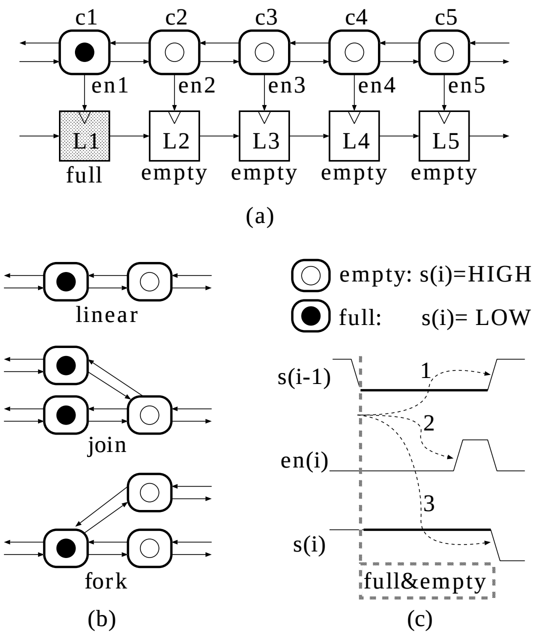

Figure 3. asP* based Asynchronous FIFO

4. Multi-Clocked Scan Register Controller

We employ **asP*** for multi-clocked scan register (MCSR). MCSR is shown in fig.4. Each of the rectangular blocks labeled $i, j (1 \leq i \leq m, 0 \leq j \leq n)$ represents scan-latch $SL_{(i,j)}$.Each $SL_{(i,j)}$ is controlled by clk in normal mode. The test mode and the normal mode are selected by sw which is omitted in fig.4.

All of them are serially connected in a string, forming a scan-path from $SL_{(1,0)}$ to $SL_{(m,n)}$. For $1 \leq i \leq m, 0 \leq j \leq n-1$, the output of $SL_{(i,j)}$ is connected with the input of $SL_{(i,j+1)}$, and for $1 \leq i \leq m-1$, the output of $SL_{(i,n)}$ is connected with the input of $SL_{(i+1,0)}$. All of $c_j (0 \leq j \leq n)$ are made in a loop which have *join* and *fork* at c_0 and c_n, respectively.

Note, the condition of en_0 producing pulse signal is only when the indications of c_{in} and c_n are *full* and the indications of c_0 and c_{out} are *empty*.

c_j controls the jth row which have $SL_{(i,j)} (1 \leq i \leq m)$. The square labeled CUT includes $m \cdot n$ scan-latches, $SL_{(i,j)} (1 \leq i \leq m, 1 \leq j \leq m)$. The 0th row of

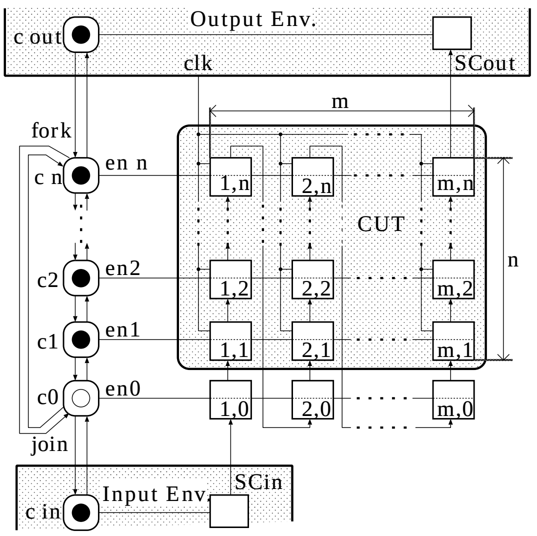

Figure 4. Multi-Clocked Scan Register with asP*

$SL_{(i,0)} (1 \leq i \leq m)$ seems to be redundant for test of CUT, but is required for shifting operation in this model.

4.1. Basic Operation

The operations of MCSR are explained as follows.

- Initialization
  - The indications of c_{out} and of c_{in} are set to *empty*.
  - The indication of c_0 is set to *empty* and the rest of controllers, $c_j (1 \leq j \leq n)$ are set to *full*.
  - sw selects the test mode.

At the initialization, $c_j (1 \leq j \leq n)$ indicate *full*, but any meaningful data are not inserted at the jth row. After the initialization, the operation of test vector insertion can start.

- Test vector insertion
  - Each bit of the test vector is serially inserted from SC_{in} to SL_0 accordingly with **asP*** communication between c_n and c_0.
  - Those inserting operations continue until the last bit of test vector is inserted from SC_{in} to $SL_{(1,0)}$.

In the operation of test vector insertion, each bit of test vector can be inserted through following processes.

1 The data of SC_{in} is updated, and c_{in} indicates *full*.

2 The conditions of *full-empty* are established between c_{in} and c_0, and between c_n and c_0, therefore data transmit from SC_{in} to $SL_{(1,0)}$ and from $SC_{(i,n)}$ to $SC_{(i+1,0)}$ $(1 \leq i \leq m-1)$. As results, the indication of c_0 changes to *full* and the indications of c_{in} and c_n change to *empty*.

3 The condition of *full-empty* is established between $c_{(n-1)}$ and c_n and data transmit from $c_{(n-1)}$ to c_n. As results, states of $c_{(n-1)}$ and c_n change to *empty* and *full* respectively.

4 Like that, data transmission between $c_{(i-1)}$ and c_i occurs, when the condition of *full-empty* is established, and next data transmission occurs between the next preceding stages.

5 When the data transmission between c_0 and c_1 occurs and finishes, the indication of c_0 changes to *empty* again.

Through the above 5 processes, the data array in the scan-path from SC_{in} to SC_{out} shift forward by a bit, and next bit of test vector can be inserted as long as the data of SC_{in} is updated and the indication of c_{in} changes to *full*.

To insert $m \cdot n$ bit of test vector, those 5 processes have to be executed $m \cdot n$ times, and after insertion of the last bit of the test vector, all bit of the test vector are serially stored in $SL_{(i,j)}$ $(1 \leq i \leq m, 1 \leq j \leq n)$. Then those bit of test vector can be evaluated, through the operation of test vector evaluation.

- Test Vector Evaluation
  - sw selects the normal mode.
  - Test Vector is evaluated by single pulse of clk.
  - After the evaluation, sw selects the test mode.
- Test vector extraction
  - The indication of c_{in} is fixed at *full*, while the indication of c_{out} fixed at *empty* is free.
  - Each bit of the test vector is serially extracted from $SL_{(m,n)}$ to Sc_{out} accordingly with **asP*** based communication between c_n and c_{out}.
  - Those extracting operations continue until all bits of test vector are extracted from all the scan-latches in CUT.

The extracting operation is almost same as inserting one. Every time *empty* rounds and arrives at c_0, a bit of the test vector can be extracted to Sc_{out} accordingly with **asP*** based communication between c_n and c_{out}.

The design impact of MCSR is independent of the number of scan-paths in the CUT, but dependent on m, n.

Next sub-section, we will examine the area impact of MCSR related to m, n.

4.2. Area Estimation

In this section, we discuss area impact of a scan path controlled by MCSR in comparison with typical synchronous one.

First, we define **lt** as an unit of occupied are for a single latch. Therefore, occupied areas of L1L2 based scan-latch(fig.1(a)) and of single scan-latch(fig.1(b)) correspond to 2[**lt**] and 1[**lt**], respectively. The circuits complexity of a single stage of **asP*** controller, c_i corresponds that of 3latches in logic gate level [5, 7]. So, here we assume that occupied area of c_i corresponds to 3[**lt**].

Assume given CUT have N scan-latches. If these scan-latches consist conventional L1L2 based scan-latches and are controlled by typical synchronous signal, the estimated area overhead $O_{typical}$ is,

$$O_{typical} = m \cdot n[\mathbf{lt}] \tag{1}$$

Assuming $m \cdot n = N$, if these scan-latches consist single scan-latches and are controlled by MCSR, the estimated area overhead O_{MCSR} is,

$$O_{MCSR} = m + 3(n+1)[\mathbf{lt}] \tag{2}$$

With simple mathematical calculations, following an inequality can be gained from (2).

$$O_{MCSR} \geq 2\sqrt{3m \cdot n} + 3[\mathbf{lt}] \tag{3}$$

The lower bound of O_{MCSR} is established, only when $m = 3n$.

From (3) it is clear that $\frac{O_{MCSR}}{O_{typical}}$ can decrease with the increase of $m \cdot n$. If the number of scan-latches controlled by MCSR is more than 2^9, O_{MCSR} can be less than 20% of $O_{typical}$.

4.3. HSPICE Simulation

We designed circuits of MCSR with scan-latches(fig.4) in CMOS level for HSPICE simulation to estimate Cycletime. For this simulation, $0.13\mu m$ technologies(VDD=1.2V) are used. As **asP*** controller, we employ GasP[7] circuits, whose CMOS schematic design is smartly optimized to improve their switching speed.

This simulation model has looped scan path connecting SCout to SCin. Initially, the scan register is written by the series of $\{10101010...\}$, so that every time vector shifts, all enabled scan-latches could make transition. Waveforms in fig.5 represent data transmissions of MCSR, the 16th stage, the 15th stage, the 14th stage, and the 0th stage, where m=32 and n=16. The waveforms of `v(en16)` and `v(17)` represent en_{16} and output of $SL_{(32,16)}$ respectively. The waveforms of `v(en15)` and `v(16)` are en_{15} and output

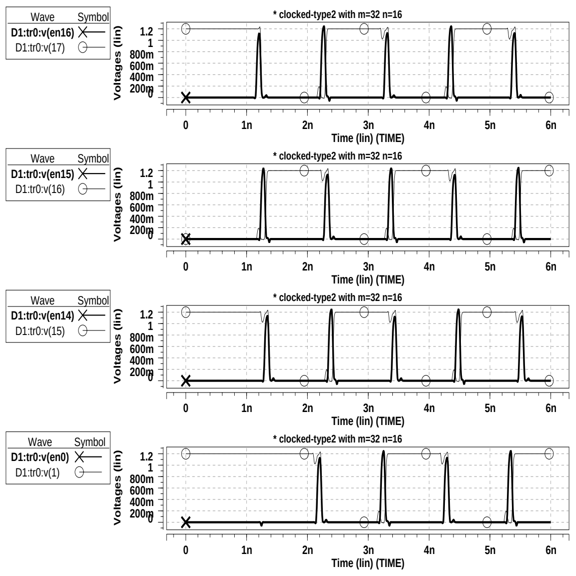

Figure 5. Wave form of MCSR for $m = 32, n = 16$

n		4	8	16	32	64
cycletime	[**ns**]	0.305	0.550	1.04	2.02	3.97
frequency	[MHz]	3,278	1,818	961	495	252
$\frac{O_{MCSR}}{O_{typical}}$	[%]	36.7	23	16.2	12.8	11.1

Table 1. Cycletime of MCSR

of $SL_{(32,15)}$ respectively. The waveforms of `v(en14)` and `v(15)` are en_{14} and output of $SL_{(32,14)}$ respectively. The waveforms of `v(en0)` and `v(1)` are en_0 and output of $SL_{(1,0)}$ respectively.

The period of cycletime of MCSR is strongly depend on n, scale of looped asP* controller. We made simulation of MCSR for different size of n with $m = 32$. Table 1 shows their results.

According to ITRS2003[1], those results promise that MCSR can not be the bottleneck of the test speed. Moreover, operation speed of MCSR is in proportion to CMOS technology rule. So, this correlation continues in future.

5. Clock-based DFT Structures employing MCSR

We propose asynchronous scan-latch controller, MCSR for lower area. Though this is based on asynchronous timing, it can accept synchronous signal and to be applied for

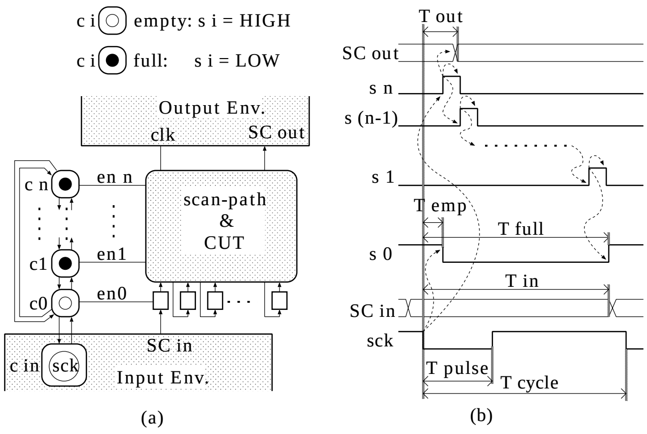

Figure 6. Clocked MCSR: s_i is internal signal of c_i

conventional DFT systems. In this section, we exhibit a clocked DFT structure employing MCSR, as an example.

5.1. Clocked MCSR

The clocked MCSR controllers is shown in fig.6(a). Comparing to fig.4, c_{out} is removed which makes communication between CUT and Output environment, and external clock signal sck emulates internal signal s_{in} to control MCSR with a synchronous signal.

Through a clock cycle of sck, MCSR generates control signal en_i $(0 \leq i \leq n)$ to shift test vector by a bit in the scan-path of CUT. The timing chart is shown in fig.6(b). T_{out} is a period from a fall of sck to an update of SC_{out}. T_{in} is a period from a fall of sck to the next change of SC_{in}. T_{emp} is a period from a fall of sck to fall of s_0. T_{full} is a period from a fall of sck to the rise again of s_0. T_{pulse} is the period of low level of sck. T_{cycle} is the clock cycle of sck. For correct operations, clock signal sck must be satisfied following inequalities.

$$T_{emp} < T_{pulse} < T_{full} \quad (4)$$
$$T_{full} < T_{cycle} \quad (5)$$

For such given clock period of sck, the timing of SC_{in} must be satisfied following inequality.

$$T_{emp} < T_{in} + T_{inHOLD} < T_{cycle} \quad (6)$$

Under those inequalities, it is assured that before the rise of sck, SC_{in} and SC_{out} are ready to be read.

T_{full} which may be the lower bound of T_{cycle} depends on n of MCSR.

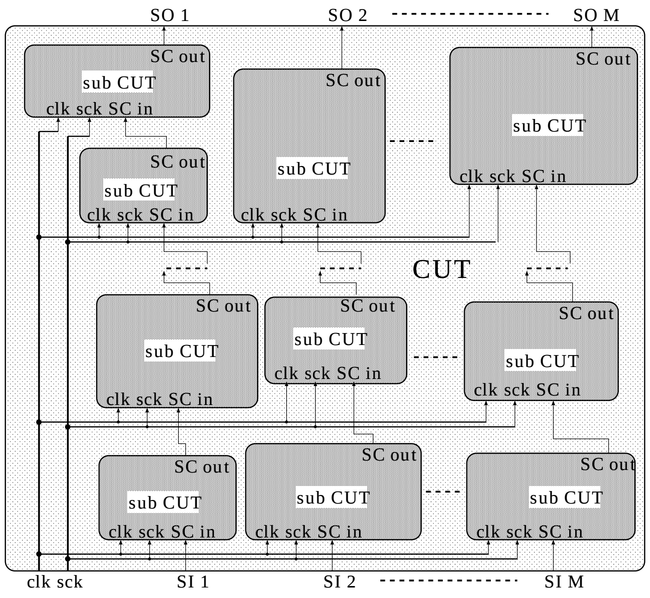

Figure 7. Application of Clocked-MCSR to local sub-circuits

5.2. Application of clocked-MCSR to local sub-circuits

The clocked MCSR can also be applied for test of local sub-circuits. Fig.7 shows an example. CUT has M scan-paths from SI_i to SO_i ($1 \leq i \leq M$), formed by a serial connection of subCUTs. Each subCUT is controlled by clocked-MCSR. Each of scan-paths can have different number and size of subCUTs, as long as sck satisfies following condition,

$$T_{out}^{Max} < T_{pulse} < T_{full}^{min} \tag{7}$$

$$T_{full}^{Max} < T_{cycle} \tag{8}$$

Here, T_{out}^{Max} and T_{full}^{Max} are the maximum periods of T_{out}s and of T_{full}s in the set of subCUT respectively, and T_{full}^{min} is the minimum period of T_{full}s.

The merit of fig.6(c) is that each enable wire en_i of MCSR avoid to form long global wire traversing the chip die of CUT, which may impact on the area overhead.

Designers can decide the region and the size of each subCUT freely as long as the total area impact of each MCSR which depends on $m \cdot n$ is not dominant in CUT.

6. Conclusions

In this paper, we propose locally controlled scan-latch register to restrain area overhead. In this model, its area impact depends only on the size of scan-paths. We made mathematical estimations and see the area impacts can be negligible under practical number and size of scan-paths.

HSPICE simulation shows the speed of shift operation of our suggested type is practical, and its trend can continue in future under prediction of ITRS2003.

We also show that our suggested asynchronous scan-latch controller can be applied to conventional clocked scan system.

7. Acknowledgments

This work is supported by VLSI Design and Education Center(VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

References

[1] *International Technology Roadmap for Semiconductors 2003* `http://public.itrs.net/`.

[2] S. DasGupta, P. Goel, R. G. Walter, and T. W. Williams. A VARIATION OF LSSD AND ITS IMPLICATIONS ON DESIGN AND TEST PATTERN GENERATION IN VLSI. *Proceedings of International Test Conference*, pages 63–66, November 1982.

[3] S. B. Furber and p. Day. Four-phase micropipeline latch control circuits. *IEEE Trans. on VLSI Systems*, 4:247–253, June 1994.

[4] D. Josephson, Don, S. Poehlman, V. Govan, and C. Mumford. Test methodology for the McKinley processor. *Proceedings of ITC 2001*, pages 578–585, 2001.

[5] C. E. Molnar, I. W. Jones, J. K. Coates, W S Lexau, S. M. Fairbanks, and I. E. Sutherland. Two FIFO ring performance experiments. *Proceedings of the IEEE*, 87:297–307, February 1999.

[6] S. Mourad and Y. Zorian. *Princibles of Testing Electronic Systems*. Jhon Wiley & Sons, Inc, 2000.

[7] I. V. Sutherland and S. Fairbanks. GasP: A minimal FIFO control. *Proceedings of Int. Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 46–53, 2001.