LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

A Design Strategy for Testable Self-timed Systems

Computation Structures Group Memo 216-1
April 1982
Revised October 1982

Tam-Anh Chu

This research was supported by the National Science Foundation under grant number MCS-7915255.

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

Computation Structures Group Memo 216
Revised October 1982

A Design Strategy for Testable Self-Timed Systems[1]

Tam-Anh Chu
MIT Laboratory for Computer Science
Cambridge Massachusetts 02139

Abstract

This paper presents a strategy for designing testable self-timed systems, using additional layout rules and hardware which could facilitate testing of the fabricated IC chips. It is assumed that a fabricated chip contains no logical faults, and that the types of physical faults present can only be stuck-at and bridging faults. The design strategy consists of a number of techniques aiming at the testability of the system. First, additional layout rules are proposed to reduce the set of physical faults to stuck-at faults alone, then test hardware is introduced to allow testing for stuck-at faults in self-timed modules. C-elements with special test modes are used for constructing self-timed combinational logic modules. In test modes, they can be configured as either AND or OR gates; thus, a self-timed combinational logic module can be transformed into two simple and related networks, from which test vectors can be derived. For self-timed state machines, scan-in/scan-out shift registers are used to test their combinational logic part and feedback paths. In test modes, the self-timed communication protocol is not observed, therefore, these techniques are used for off-line testing only.

---

1. This research was supported by the National Science Foundation under grant number MCS-7915255.

A Design Strategy for Testable Self-Timed Systems[1]

1. Introduction

This paper presents a strategy for designing self-timed systems, using additional layout rules and hardware which could facilitate testing of the fabricated IC chips. First, a high level design hierarchy including functional, logic simulations, and mask generation, etc. is suggested. This will allow us to argue for the assumptions made on the types of faults a VLSI self-timed system might have. Thereafter, we will present a number of techniques permitting design for testability of VLSI self-timed systems.

2. High Level Design Hierarchy

The block diagram in Figure 1 shows a flow chart of the steps required in the design of self-timed VLSI chips, up to the level of mask generation. Associated with each step is an appropriate computer aided design tools (shown in small boxes), including simulation softwares, artwork description language and design rule checker. The following paragraph describe briefly each step in the flow chart.

2.1 Functional Specification and Simulation

Functional specification and simulation can be considered as the most important steps in the whole design process. Here, the specification of the system is generated from a high-level hardware description language such as PADL[7]. The functional simulator takes these descriptions and simulate the behavior of the system and its submodules. During this phase of simulation, problems with system design, including deadlocks and logical errors should be detected and corrected.

---

1. This research was supported by the National Science Foundation under grant number MCS-7915255.

2.2 Circuit Simulation

A transistor level simulator such as SPICE [13] can be used to aid the design of standard cells, in our case, self-timed modules. It allows the optimization of the designs in terms of delay, power dissipation, and also allows one to check the timing behavior and ensure that they do not produce output hazards due to independent input changes. Once the cells are verified, they can be used without any unanticipated timing problems at the system level.

2.3 Mask Description

The transistor network layout can be described using layout description languages. A symbolic description language such as Design Procedural Language [2] (DPL) can describe the chip by specifying the relative locations of the standard cells (which are descriptions of transistor circuits) and their interconnects..section "Logic Simulation" A gate level simulator using binary or ternary values can simulate a large network, such as the whole chip or a subsystem. For MOS networks, a particularly useful logic simulator is MOSSIM [3], which models the MOS transistor as a bidirectional switch with charge holding capacitances, and uses ternary representation of logic levels. As shown in the block diagram, the input to the simulator includes information from the functional specification, and the transistor network description that can be extracted from the mask layout description by a node extraction program [1]. It should be emphasized that the purpose of logic simulation is to verify that the translation process from functional specification to transistor networks is correctly carried out. Whereas in other systems using synchronous design or bus architecture, timing simulation and verification are of utmost importance, in self-timed system design, it only serves the purpose of speed optimization, and logic simulation alone would be adequate for verifying the design. The reason is that for self-timed systems, inter-module or long distant communication is implemented using the reset signaling protocol [11], which is speed-independent and therefore, free of timing skews and races. At the local level, timing constraints of each self-timed module can be satisfied easily by using a standard set of macro cells whose electrical and timing characteristics have been verified by a circuit simulator, as mentioned above.

2.5 Design Rule Checking

The design rule checker checks for design rule violation in the layout. A set of simple rules (Mead and Conway's) can be implemented, and a large portion of the layout description can be input to the checker.

The design process usually takes a number of iterations before the design passes both the simulation and design rule checking steps. At this point the mask description can be forwarded for fabrication.

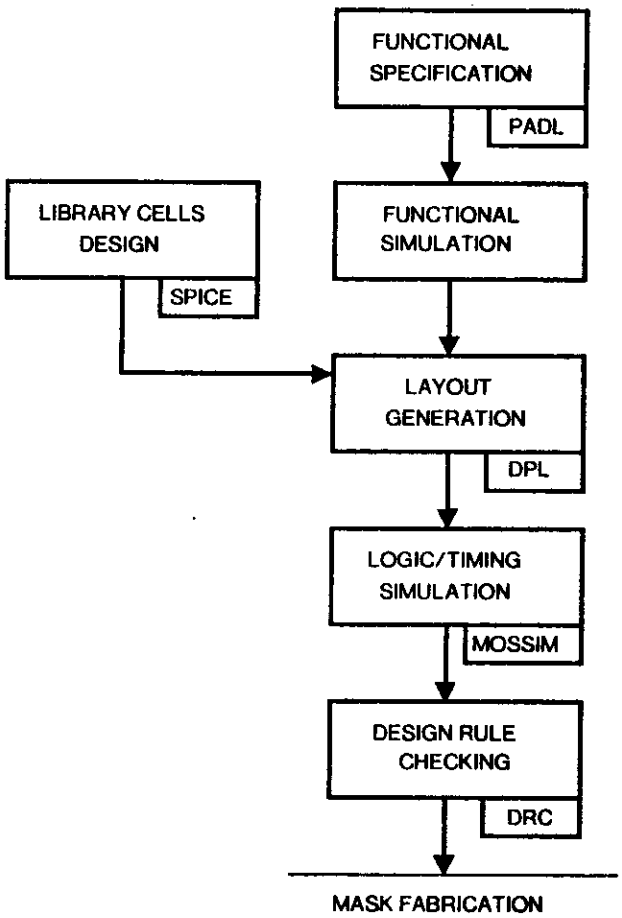


Figure 1. Flow chart of a Self-Timed VLSI design system

3. Assumptions on Types of Faults in VLSI Self-Timed Systems

As described in the previous section, a VLSI self-timed system will only be fabricated after it has been proven that the system is functionally correct and free of logical errors. This is done by functional simulation, mask information extraction, and logical simulation. These steps guarantee that the transistor networks generated behave according to the functional specifications of the system. Thus, the types of faults existing at the chip level could only be physical faults, i.e., those caused by defects, process variations, etc.

We will assume two main types of physical faults that can affect the circuits.

3.1 Stuck-at Faults

This type of fault modeling is most popular for fault simulations of digital systems. Stuck-at-1 and Stuck-at-0 faults of a logic gate correspond to cases when its input or output signal paths are shorted to the power supply voltage or to ground, respectively. A great deal of work and many strategies developed for testing systems with stuck-at faults can be found in the literature.

In a self-timed systems, stuck-at faults may affect either the handshake signals or the data signals. In the first case, the system stops running because the effect of the faulty communication signal will eventually propagate to the system input and output ports. In the latter case, data integrity is no longer preserved. If dual-rail coded data were used, the system would always 'hang', as the data signals themselves implicitly contain a request signal.

3.2 Bridging Faults

In many cases, the stuck-at fault model does not adequately describe many real situation caused by faults in VLSI systems. In fact, one can consider stuck-at fault as a special case of bridging fault, occurred due to shorts between signal lines. The stuck-at fault model, which originally came from fault modeling in DTL digital circuits, represents all faults as shorts between signal paths and power supply busses (Vdd and ground). In a piece of combinational logic circuit, if some of its input and output signal paths are shorted, an asynchronous state machine can accidentally be formed. If pull-ups and pull-downs are shorted, analog voltage levels might result. In other cases, shorts between signal lines can cause stuck-at faults.

Because of all these peculiarities introduced by shorted signal paths, bridging faults are generally much harder to detect and diagnose compared to stuck-at faults. For example, an analog voltage caused by pull-up/pull-down shorts can manifest as stuck-at-one, stuck-at zero, an intermittent fault, or it might show up at the system output as an analog signal.

As discussed in [6], experimental results indicated that bridging faults are very common in VLSI chips. Bridges occur almost exclusively between signal paths at the same layer of interconnects, i.e., between two metal lines, two diffusion lines, etc. Another source of bridging faults is due to shorts between metal lines and the substrate.

Another type of faults complementary to bridging faults should be mentioned here : faults due to broken signal paths, causing opens. In static operation, these faults can be modeled as stuck-at faults because an unconnected signal line will eventually discharge to ground. In a VLSI chip, these faults are commonly caused by missing contact cuts, defective chip areas. Metal interconnects and busses suffer most due to the fact that they run on a rugged terrain and carry high currents, a phenomenon known as metal migration can break off metal lines and thus cause open-circuits.

4. Design for Testability Strategies

Having discussed a number of fault models for VLSI systems, we now propose a number of design techniques aimed at enhancing system testability. The first technique deals with bridging faults, and may be applicable to VLSI system in general; the approach taken is an extension and modification of that discussed in [6], in which additional layout rules are put forth to minimize the probability of shorts in the digital circuits. The second technique is more specifically derived for testing self-timed systems. At the hardware level, additional test capability is designed into self-timed modules such that enough *controllability* and *observability* are introduced for testing of stuck-at faults.

4.1 Layout Approach for Reduction of Bridging Faults

As discussed earlier, bridging faults can cause anomalous behaviors, and it is difficult to detect as well as diagnose the faulty modules in this case. In this section, we briefly summarize the sources of bridging faults and their effects. Then, a layout rule (Mead and Conway's) is evaluated in terms of its susceptibility to bridging faults. Next, interconnect models at logic gate and block levels are introduced, which lead to a definition of a set of layout guidelines for reducing bridging faults.

4.1.1 Sources and Effects of Bridging Faults

As mentioned earlier, bridging occurs between signal paths on the same layer of interconnects. In a typical NMOS process with three layers of interconnects (metal, diffusion and polysilicon), three types of shorts can occur. Shorts between diffusion and polysilicon is less likely, and are neglected for simplicity. Also, shorts from metal lines to the substrate at contact cuts between metal and diffusion are not considered because they can only be prevented by processing techniques.

Shorts between two inputs or two outputs usually create analog voltage levels or stuck-at faults, whereas shorts between inputs and outputs of a combinational logic circuit can produce an unwanted asynchronous state machine.

4.1.2 Evaluation of Mead and Conway's Layout Rules

Design rules for NMOS VLSI systems introduced in [8] form a simple and conservative set of rules for layout. They are specified in terms of the basic metric λ, being equal to the fundamental resolution of the process used. It is the *distance by which a geometrical feature on any one layer may stray from another geometrical feature on the same layer or on another layer, all processing factors considered*

and an appropriate safety factor added [8]. Following these rules, for example, two poly lines are placed at least 2 λ apart to prevent them from being shorted to each other. Other rules deal with separation between other layers, and their effects are similar, that is, they prevent bridges between signal paths. Thus one can conclude that Mead and Conway's design rules are appropriate for eliminating bridging faults. However, to further ensure that bridging faults exist at a very low probability, we derive additional rules on the relative placement of elements within a logic gate, and relative placement of logic blocks within a subsystem or system. A difficult requirement for these rules is that they do not work against the previously adopted layout rules. For example, it is not useful to derive another rule which says that two poly lines should be placed at least 3 λ apart.

4.1.3 Layout Rules for Reducing Bridging Faults

Additional layout rules exist at two levels : logic gate and logic block, the former being a single logic function with one output, the latter a network of interconnected logic gates. The approach used to derive the rules is : first, define a model for the gate or block, identify the possible interactions between elements, then derive rules to inhibit cases where interactions can lead to bridges between elements.

4.1.3.1 Logic Gates

For an NMOS static logic gate, the model used in Figure 2 is adequate to describe it. The logic gate contain a pull-up (typically a depletion load device), one output, and a network of inputs consisting of one or more conduction paths, with one or more transistors on each path. A conduction path is defined as a path connecting the output to ground when all transistors on that path are turned on. Given that our typical process implements conduction paths on diffusion, inputs on poly layer, outputs on poly or metal, possible types of shorts are: input-input, input-output, and conduction path-conduction path. Those between conduction paths and inputs or output are less likely to occur. The following rules can further eliminate the shorts between inputs, and between output and inputs.

> 1. Wherever permitted, arrange the elements such that the spacing between adjacent signal lines on the same interconnect layer is as large as possible, with the constraint that the gate layout is as small as possible (see layout examples below).

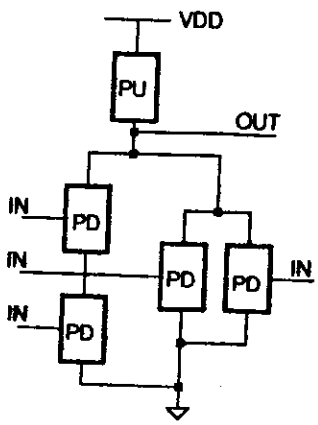


Figure 2. Model of an NMOS logic gate

2. Inputs and output are kept geographically apart by routing all inputs from one side of the logic gate, outputs from the other.

Rule 1 eliminates shorts between inputs, while 2 between inputs and outputs.

Two examples are presented below . In the first one (Figure 3), two layouts for a three input NOR gate are shown, the left one is laid out with no consideration for bridging faults, and the possible locations where bridges can form are indicated. In the layout at right, laid out to minimize the occurrence of bridging faults, contact cuts are inserted between input poly lines to increase their spacings. Note that the second lay out has all the possible short locations removed. In the second example (Figure 4), layouts for the C-element are shown, the one at left has many possible short locations indicated while the one at right has none.

4.1.3.2 Logic Blocks

A logic block is defined as a network of interconnected logic gates. In Figure 5, a logic block containing three logic gates is shown. The possible fault locations due to bridging are indicated. The following two rules can be implemented to reduce bridging faults.

1. Logic gates should be placed so that loops can not be formed, that is, output should not be routed back to the input side of the logic gate.

2. In order to eliminate the interactions between elements internal and external to a gate, a safety boundary should be included in the layout of the gate.

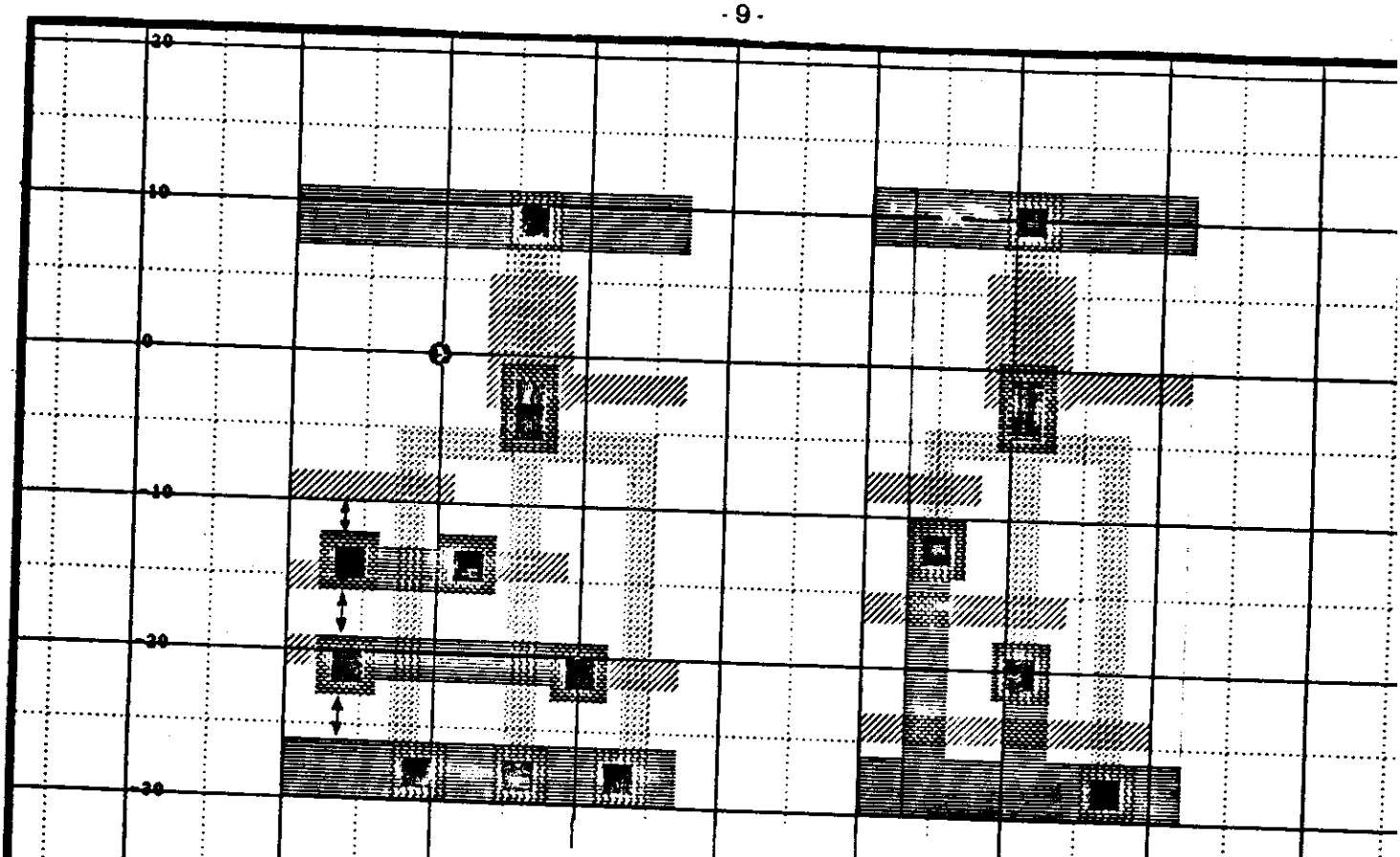


Figure 3. Layouts of three-input NOR gates

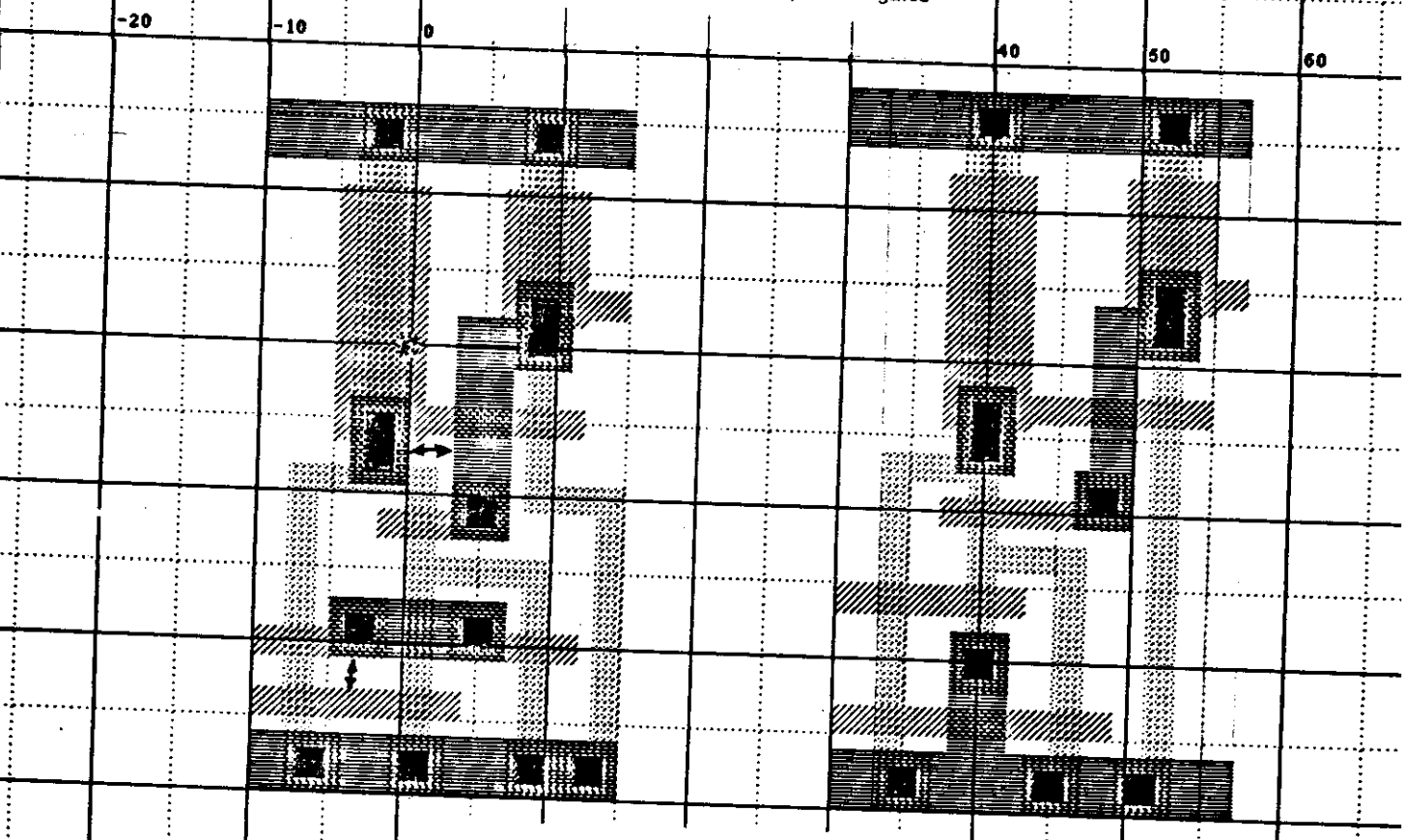


Figure 4. Layouts of the C-elements

The safety boundary is defined as the region in which no external elements can interact with internal ones. In Figure 6 below, a gate is laid out with horizontal metal busses. The verticals strips and the busses form a safety boundary. The only thing allowed in it is input or output interconnects. Note that the modified layouts in Figures 3 and 4 contain safety boundaries and therefore are slightly larger than those on the left.

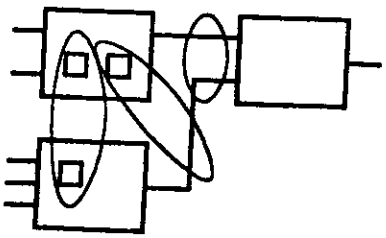
4.1.4 Hardware Approach to Stuck-at Faults

If the above layout rules are strictly applied to designing VLSI systems, one can be much more ascertained that bridging faults seldom occur, and the stuck-at fault assumption holds true for such systems. Since there has been a great deal of work done on test pattern generation for combinational logic networks under stuck-at fault assumption, we will only be concerned about designing hardware to allow easy testing of such networks. In this section, we first discuss the strategy used for designing testable self-timed VLSI systems. Then we developed in detail the design techniques for self-timed combinational logic and self-timed state machine. Occasionally, specific self-timed hardware used in the implementation of a self-timed router module [10] will be drawn upon as examples.

4.1.4.1 General Strategy

4.1.4.1.1 Network of Self-Timed Modules

We define a self-timed module as a circuit whose input/output timing specifications follow the weak-conditions and an asynchronous communication protocol [11]. A primary self-timed module is one which can not be further decomposed into other self-timed modules, thus is the lowest-level entity of self-timed circuits. A secondary self-timed module is one composed of more than one primary module. A self-timed system is a network of interconnected self-timed modules, primary or



Large square- Logic gates
Small square - internal elements

Figure 5. Model of a Logic Block

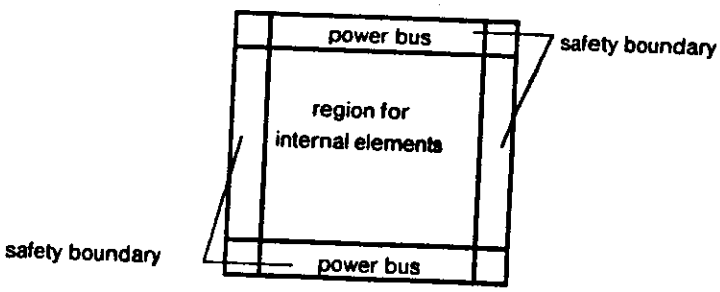


Figure 6. Layout constraint for a logic gate

secondary, as shown in Figure 7. According to this definition, it is absolutely arbitrary to define the boundary of a self-timed module, and our partition is mainly for defining appropriate boundaries to incorporate test circuits. In Figure 7, each communication link between modules consists of a tuple of data lines and a Request/Acknowledge signal pair required for implementing the reset signaling protocol.

A self-timed combinational logic (STCL) module is one with no state information retained, thus can either be a primary or secondary self-timed module. One way of synthesizing STCL using dual-rail coded signals and C-elements is shown in Figure 8. A self-timed state machine (STSM) is one with state information stored in feedback registers. Thus it is necessarily a secondary self-timed modules. Figure 9 shows an implementation of a STSM , using Join, Fork , Register modules and a STCL module [12]. These have been used in [10].

4.1.4.1.2 Design for Testability Strategy

Two key concepts utilized in our design strategy are those of *divide and conquer* and *increase controllability and observability*. These are simple but powerful concepts used in the field of design for testability [9]. The strategy can be briefly described as follows

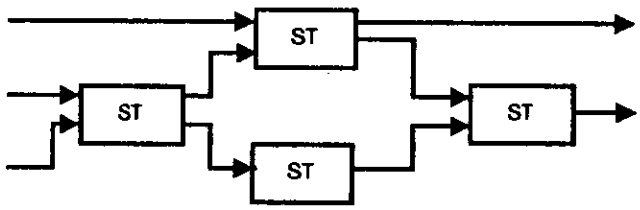


Figure 7. A network of Self-timed modules

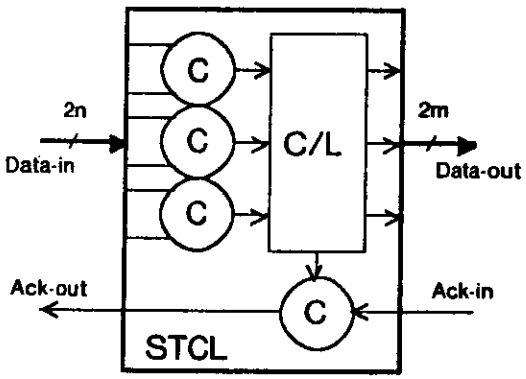


Figure 8. A Self-Timed Combinational Logic Module

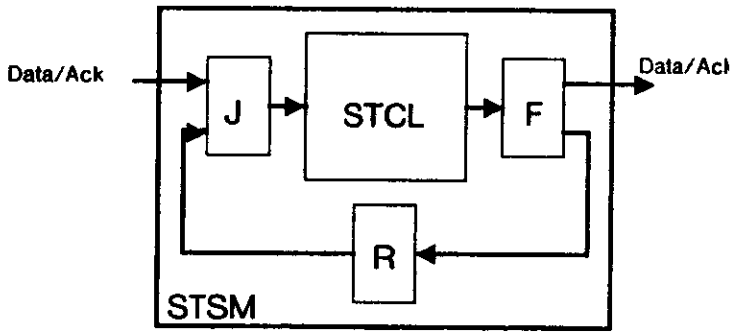


Figure 9. A Self-Timed State Machine

1. If a self-timed system consists only of STCL's, it is transformed into a single combinational network with all timing restrictions removed. This is done by setting all C-elements in Test mode.

2. If a self-timed system consists only of STSM's (not a likely case).a different technique is used, in which, shift registers are inserted at the input and output of the STCL of the state machine. Next the rest of the circuit is isolated and test vectors and results can be shifted in and out through an additional serial line for testing purpose. This technique is similar to a host of others described in [5,9].

3. If a self-timed system consists of both types of module, then both techniques proposed can be used. Note that in this case, the combinational and the sequential parts are separated by the isolation shift registers.

The point to be stressed here is that in test mode, the system is transformed into a different network which allows for testing of mainly stuck-at faults. This strategy is used only for off-line testing.

4.1.4.2 Combinational Logic Networks

4.1.4.2.1 Concept

In a STCL module, C-elements are used to synchronize signal events and enforce the timing constraints of the reset signaling protocol. The C-element can be designed with test modes, in which it behaves either like an AND or an OR gate, depending on the test modes desired. In the AND test mode, it allows selective control of signals, while in OR test mode, it allows signal to feed through. Thus, a STCL network can be transformed into two test networks, providing more information for testing. Furthermore, additional flexibility is obtained by individually programming the self-timed modules in AND or OR test mode.

4.1.4.2.2 Implementation

Because the basic structure of the C-element already contains AND and OR logic function, it is very easy to implement the test modes. Figure 10 shows a block diagram of a C-element with test modes, where a Multiplexor is used in the feedback path. It is selected by the test signal TEST and the test mode is defined by the external input OR. The table lists the functions of the C-element depending on the test mode settings. In the NMOS implementation, a multiplexor can be built out of pass transistors and thus the layout area does not increase substantially. Also, the delay through the pass transistor is very small, introducing only little additional delay in the feedback path. SPICE simulations show that

its performance is almost the same as that of one with no extra feedback delay.

4.1.4.2.3 Discussion

The test modes of the C-element greatly facilitate testing of self-timed circuits. As the C-element contains AND and OR gates, by breaking the feedback path and setting the feedback input at either zero or one, we can test the AND or the OR gates of the C-element. Test generation for such a circuit is also greatly simplified : because each C-element can be configured as an AND or an OR gate, a few test vectors can totally check a C-element for stuck-at faults.

Note that there is no equivalent technique for synchronous systems, as the latches used in a clocked system do not have simple structures for incorporating test modes.

4.1.4.3 State Machines

4.1.4.3.1 Concept

The technique used for testing STSM's is to insert shift registers at the input and output ports of the combinational logic part, isolate it from the rest of the system, and shift bit strings in and out serially. The block diagram of the circuit is shown in Figure 11, the Join and Fork modules are implemented simply as wires and C-elements. In test mode, the feedback registers will feed data through, i. e., their input and output ports are shorted together. The shift registers have a number of modes : Shift, Load and Hold. In order to test the combinational logic, a test vector is shifted in serially. After waiting a

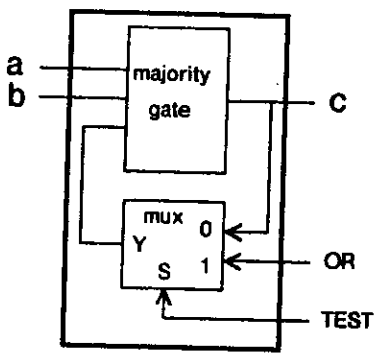


Figure 10. Implementation of a C-element with test modes

TEST	OR	FUNCTION
0	X	C-element
1	0	AND-gate
1	1	OR-gate

Test modes of the C-element

number of test clock cycles for propagation delay through the combinational logic network, its output is loaded into the output shift registers, while whatever was previously stored in the output shift registers is loaded into the input shift registers. The content is then shifted out serially. Thus one can check for stuck-at faults in both the combination logic and the feedback registers.

4.1.4.3.2 Implementation

The block diagram of the shift registers and their implementation in NMOS are presented in Figures 12(a) and (b). Each register needs four control signals which are derived from the external test input signals SHIFT, LOAD, TEST and the test clock Φ. Internally, Φ is used to generate two non-overlapped clock phases Φ1 and Φ2. More logic is used to convert them into the required signals, as shown in Figure 12(c).

4.1.4.3.3 Discussion

The above technique for testing state machines can be easily incorporated into a state machine implemented in Programmable Logic Array (PLA) form. A PLA requires input and output buffers to drive large capacitance loads on the AND plane and OR plane busses. A little extra hardware is needed to convert these buffers into isolation shift registers. On the other hand, techniques used in synchronous systems such as Level Sensitive Scan Design (LSSD) [4] or Scan Path , require a substantial amount of additional hardware to implement the test registers. Also, in the techniques proposed above, the number of extra wires dedicated for the test circuits is comparable to that required by others (LSSD requires 3 wires for the test registers).

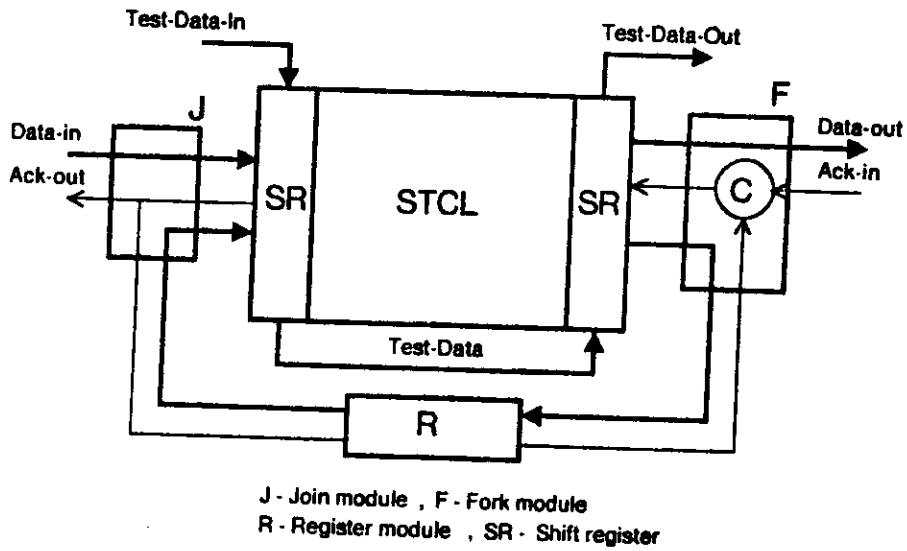


Figure 11. Implementation of test hardware for

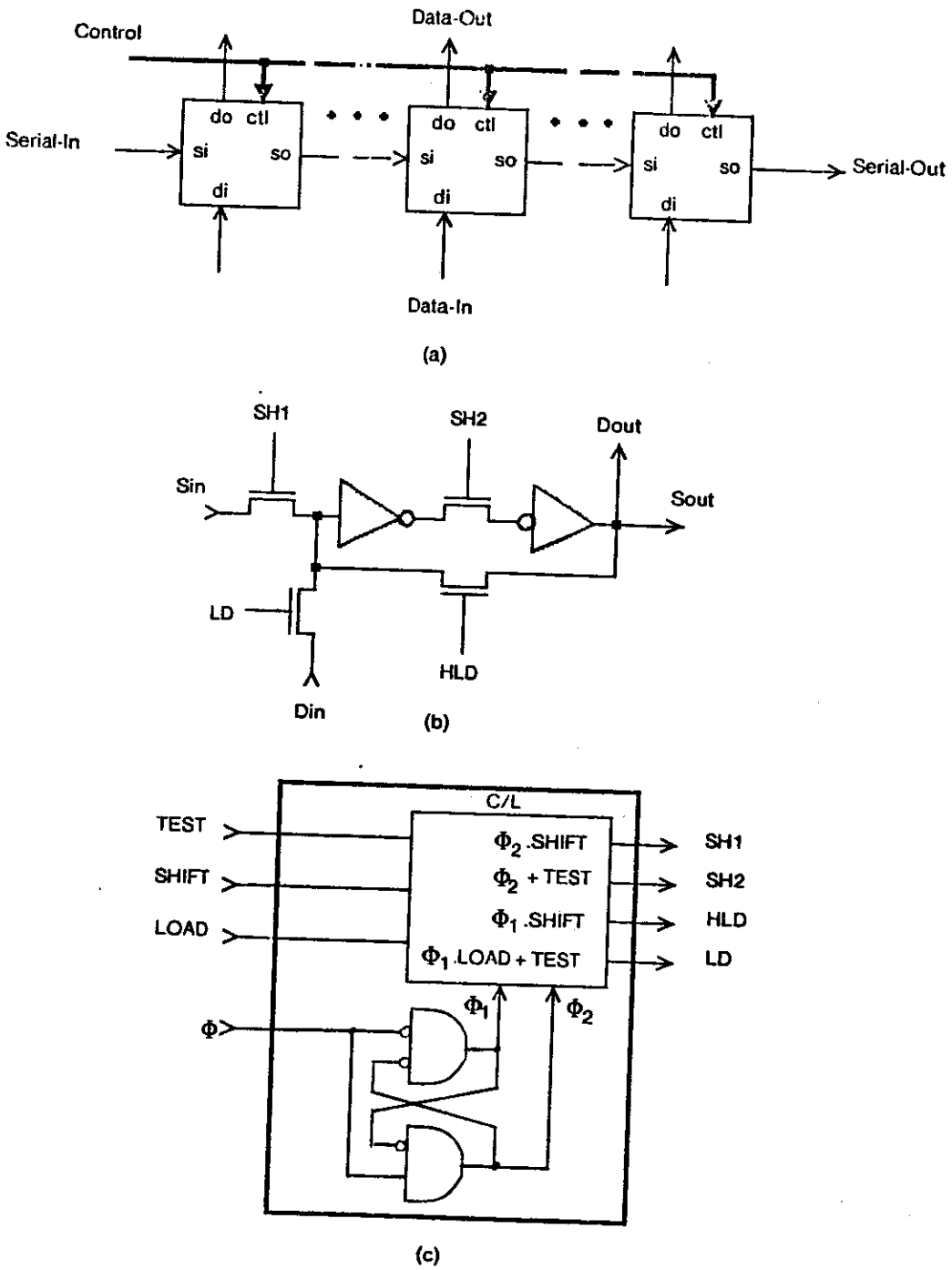


Figure 12
(a) Block diagram of a shift register
(b) NMOS implementation of a shift register cell
(c) Control logic for the shift register

5. Conclusion

This paper presents a strategy for designing testable self-timed systems. The types of faults in the system can be either stuck-at or bridging faults, as it is assumed that the high-level design and simulation steps can guarantee a system free of logical errors. Since bridging faults are complicated to deal with, additional layout rules are introduced in order to minimize their occurrence. At this stage, only stuck-at faults prevail. Two techniques are employed for testing self-timed combinational logic circuits and self-timed state machines. Exploiting the structure of the C-element, we incorporate test modes that can configure C-elements as AND or OR gates. The result is one can obtain two combinational logic networks for testing, thus reducing substantially the number of test vectors required for testing stuck-at faults. The second technique for testing state machines is similar to others like LSSD, etc., it allows testing for stuck-at faults of the combinational logic and feedback paths of a state machine.

An issue of considerable interest which has not been addressed in this paper is the hardware technique allowing on-line testing of self-timed systems. In a self-timed system, each self-timed module interacts with its neighbors by Request/Acknowledge signal pairs, and it asserts them at a rate determined by internal delays of the module. By holding back the assertion of the acknowledge signal of the communication lines, one can temporarily stops the system to perform some on-line testing. Normal operation resumes once the acknowledge signal is released. This is analogous to 'stopping the clock' in a synchronous system, but is much more flexible and localized. A strategy needed to realize this technique can be quite involved, and the hardware required can be substantial; however, the potential advantages for testing self-timed systems should make this approach favorable.

6. Acknowledgments

I would like to thank Prof. Jack Dennis for his encouragement, support and criticism. Thanks are also due to Bill Ackerman, Andy Boughton for helpful discussions. Lastly, much insights have been gained through discussions with Willie Lim, whose interest in this subject provides a strong motivation for me.

6. References

1. Baker, C. M., Artwork Analysis Tools for VLSI Circuits, VLSI Memo 81-16, June 1980.

2. Batali, J. and Hartheimer, A., The Design Procedure Language Manual, VLSI Memo 80-31, September 1980.

3. Bryant,R. E., MOSSIM : A Logic Level Simulation for MOS LSI, VLSI Memo 81-47, April 1981.

4. Eichelberger, E. G., and Williams, T. W., A Logic Design Structure for LSI Testability, Proc. 14th Design Automation Conference, ACM, New York, June 1977 , pp 462-468.

5. Frank, E. H. and Sproull, R. F., Testing and Debugging Custom Integrated Circuits, Computing Surveys, Vol. 13, No. 4, Dec. 1981, pp 425-451 .

6. Galiay,J., Crouzet, Y. and Vergniault, M., Physical Versus Logical Fault Models in MOS LSI Circuits Impact on Their Testability, IEEE TC Vol C-29, No. 6, June 1980, pp 527-531.

7. Leung, C. K. C. and Lim, W. Y-P., PADL : A Packet Architecture Description Language, Preliminary Reference Manual. MIT-LCS Computation Structures Group, October 1981.

8. Mead and Conway, Introduction to VLSI Systems, Addison-Wesley, Reading Mass. 1980 .

9. Mueldorf, E. I. and Savkar, A. D.,LSI Logic Testing : An Overview, IEEE TC Vol C-30, No. 1, January 1981, pp 1-16.

10. Ries, P. S., A VLSI Implementation of a Two by Two Router, VLSI Memo 80-22, June 1980.

11. Seitz, C., System Timing, Chapter 7 of Mead and Conway.

12. Singh, N. P., A Design Methodology for Self-Timed Systems, MIT-LCS TR-258, February 1981.

13. Vladimirescu, A., Newton, R. A. and D. O. Pederson, SPICE Version 2F.1 User's Guide, Department of EECS, UC Berkeley.