Critical Hazard Free Test Generation for Asynchronous Circuits

Ajay Khoche
Sunrise Test Systems
47211, Lake View Blvd
Fremont CA, 94538, USA

Erik Brunvand
Dept. of Computer Science
University of Utah
Salt Lake City, UT - 84112, USA

Abstract

We describe a technique to generate critical hazard-free tests for self-timed control circuits build using a macro-module library, in a partial scan based DFT environment. We propose a 6 valued algebra to generate these tests which are guaranteed to be critical hazard free under an unbounded delay model. This algebra has been incorporated in a D-algorithm based automatic test pattern generator.

1 Introduction

Testing asynchronous circuits is a relatively new area compared to testing of synchronous circuits. It is also one of the obstacles in reducing asynchronous circuits to practice. Traditionally, testing asynchronous circuits has been considered a difficult problem. This is due in large part, to the absence of global clock signals. The absence of a clock, not only renders many methods for testing synchronous circuits inapplicable to asynchronous circuits, but also introduces additional problems. One such problem related to hazards and races is the possible invalidation of tests. Due to hazards and races in the circuit during testing, a test may give incorrect results even when there is no actual fault in the circuit. This is due to the fact that asynchronous storage elements used in these circuits react to all transitions, and any spurious transition during testing may result in latching incorrect data in those storage elements thus leading to incorrect outputs for a given test vector.

In this paper the issue of invalidation of tests is addressed for the control networks of a subclass of asynchronous circuits called self-timed circuits. The specific circuits targeted are those built using a library of self-timed modules called macro-modules. Macro-modular control circuits have been used in a wide variety of academic research efforts[4, 8, 12], as well as in the industrial research settings[2, 11]. A new method using a 6 valued algebra has been proposed to generate tests which are critical hazard free i.e. guaranteed to be free from being invalidated due to races and hazards under all possible delays, allowing an unbounded delay model to be used. A single stuck-at fault is assumed for generating tests.

2 Self-Timed Circuits

Self-timed circuits are a subset of a broad class of asynchronous circuits. These circuits generate completion signals to indicate that they are finished with their processing[9]. A signalling protocol used with the completion signalling allows self-timed systems to be composed of circuits which communicate using this protocol. Self-timed protocols are often defined in terms of a pair of signals, one to request or initiate an action, and another to acknowledge that the requested action has been completed. One module, the sender, sends a request event (*Req*) to another module, the receiver. Once the receiver has completed the requested action, it sends an acknowledge event (*Ack*) back to the sender to complete the transaction. The circuits in our library use two-phase transition signaling[9] for control.

2.1 Self-Timed Module Library

The specific modules used to build the control circuits considered in this paper are those described in[5, 10]. These modules are described in brief in this section and are shown symbolically in Figure 1

XOR behaves as an OR for transition signals. When a transition occurs on any of its inputs, the XOR generates a transition at its output.

C-Element is used as an AND function for transitions. A transition occurs at the output only when there have been transitions at both of its inputs.

Select module causes a transition on one of two outputs in response to an input transition, depending on the value of its select signal. The *sel* signal should be valid before the input transition arrives and must remain valid until after an output transition is generated at one of the outputs. Our Select module is designed using XORs and gated latches.

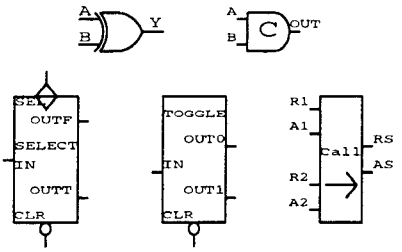

Figure 1: Control Modules

Toggle module, in response to an input transition, causes an output transition alternately on its two outputs. After initialization, the first input transition causes a transition on out0

0-8186-7810-0/97 $10.00 © 1997 IEEE

and subsequent input transitions cause transitions on alternate outputs. Our Toggle module is designed using gated latches.

Call module acts as a hardware subroutine call allowing multiple requesters to access a shared resource. The Call module routes the *Req* signal from a client (for example, either *R1* or *R2* in a two-way Call) to the subroutine circuit, and after the subroutine acknowledges, routes the *Ack* back to the appropriate client. The requests must be mutually exclusive. The Call module is made up of XORs and C-elements.

3 Test Environment

An important point to note from the previous section is that all the modules mentioned above are constructed out of XORs, C-elements and gated latches. Thus the entire control network can be viewed as an interconnection of these three modules. A partial scan solution was proposed in[6], in which only the gated latches were scanned, while the C-elements were not. This scheme results in the entire circuit being partitioned into networks of XORs/C-elements. The inputs to these partitions are either the primary inputs or inputs from the scan path. There are no cycles in the partitions except internally in the C-elements which are sequential. A static test environment is assumed where the inputs are applied through the scan path and the outputs are allowed to settle. The output is then captured in the scan path and finally scanned out. As such under an unbounded delay assumption one will have to wait for an unbounded amount of time for the outputs to settle. However a practical bound can be placed on how long the network will take to settle, based on technological specifications. A single stuck-at fault model is assumed in this test environment.

4 Test Generation Problem

Test generation for the control network under the test environment described above implies generating tests for each of the partitions consisting of XOR/C-element networks. The C-element contained in the partitions are sequential and will need a sequential test pattern generator. However, since these networks are asynchronous, conventional sequential test pattern generators for synchronous circuits can not be used. In addition, asynchronicity imposes an extra restriction on the test patterns to be free of hazards and races which can lead to steady state errors resulting in the invalidation of the test.

At present no software exists to generate such a test. One possible approach is to generate the test by functionally modeling the C-element latches without taking into account the hazard and races. This vector can then be simulated on the circuit using a hazard simulator to detect any hazards and races. A test can be rejected if it results in a potential steady state error. The problem with this approach is that if the test fails the hazard and race criteria then there is no way to rectify this test to avoid that situation, instead a different vector will have to be generated. In fact, one may not find a test even if one exists because there is no systematic search procedure for the test.

Another possible solution is to analyze the hazard behavior of each component and then use the 27 valued algebra proposed by Breuer and Harrison[3] in 1974. However this method has several practical problems as described below

- Their method was not for general asynchronous circuits, rather a synchronous environment was assumed and in that environment logic clocks and inputs to latches were considered for hazard-freeness. The storage elements were modeled functionally. The assumption about the presence of a clock allowed them to treat the inputs to the circuit as primary inputs and rely on storage elements holding their states between clocks. This is not true for asynchronous circuits where there is no clock. Especially in a scan environment described above where different values may appear at the input of a partition during scan-in operation. One will have to add a hold signal to all the storage elements to retain the circuit state during the scan-in operation.

- The second problem is related to the number of values used to keep track of hazard generation and propagation during the test generation. In their method four values are required for each line to represent various conditions. First the value of a line over two successive time frames is required to decide the hazard condition on that line. Each value could be one of the values from the set {X, 0, 1}. Another flag is required to keep track of the hazard status for hazard propagation through the elements. The hazard status flag could be one of the values from the set {no hazard, hazard present, hazard status unknown}. These values together give rise to a 27 valued logic. In addition to the three values attached to a line a fourth value is required to attach a hazard free requirement on a line during line justification. The 27 values are required just to keep track of basic logic values. The faulty values which require these values to be tracked for both the good and faulty circuits will further increase the number of values need for test generation. A larger set of values, required to keep track of various values in the circuits, implies a bigger solution space, thus the complexity of test generation will be higher.

- In their method a synchronous sequential test pattern generator was assumed where the propagation and justification of values was done across the time frames. In asynchronous circuits due to the absence of a clock the time frame assumption can not made because time is continuous.

In summary the approach proposed in[3] has problems of test generation complexity, test environment assumption and the model for test generation which make the method impractical for pure asynchronous circuits.

In order to cope up with the complexity of the state space and test generation for the asynchronous circuits of interest a circuit modification was proposed in[6] for the C-elements. C-elements are usually implemented as complex CMOS gates but for test generation a gate level model suffices. A gate level C-element model is shown in Figure 2(a) The modified C-element model is shown in Figure 2(b). Here an OR gate has been introduced in the feedback path of the C-element. This OR gate allows us to partially control the state of the feedback wire, which determines the operation of the C-element. When the state of the feedback wire is 1, the C-element acts like an OR gate. The C-element acts as an AND gate if the state of the feedback wire is 0. The purpose of the extra circuit is to be able to force a 1 on the feedback wire and force the C-element to behave like an OR gate. In this mode the circuit becomes combinational and any conventional combinational test pattern generator can be used to generate tests. However this mode of operation of does not cover all the faults in the C-elements. which are a significant portion of the total faults in the network. In order to test these faults the C-element also needs to be tested in the AND mode. The C-element can be

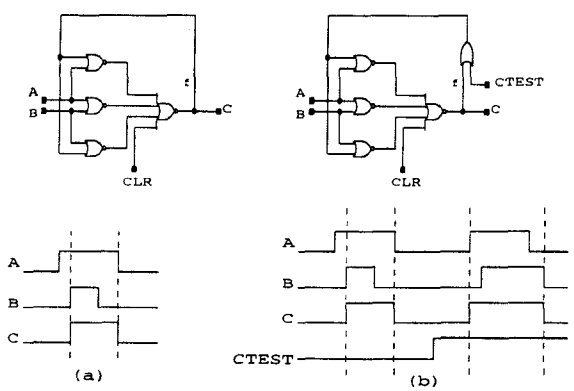

Figure 2: C-element designs

put into AND mode by asserting the CLR signal. However as soon as the CLR signal is de-asserted the C-element returns to normal mode. Now when the tests are applied it will react to all the transitions including transitions due to hazards and races. Thus testing in this mode still requires generation of critical hazard free tests. This is exactly the problem addressed in the remainder of this paper. The tests generated using the proposed method are guaranteed to be free of critical hazards i.e. will not lead to any steady state errors under any possible delay assignments in the network. Also the solution space is systematically searched such that the test is guaranteed to be found if it exists. The proposed modification allows such tests to be generated in a combinational way i.e. one does not need to keep track of history of values on a line.

5 Method for Hazard Free Tests in AND mode

In order to generate critical hazard free tests one needs to monitor the generation and propagation of hazard values. The proposed method, which needs to generate tests only for the C-elements in the AND mode, results in a reduction in the number of state transitions one needs to keep track of. By restricting the initial state of all the storage elements to 0, all the state transitions in the absence of hazards are monotonic i.e. The C-element either remains in the 0 state or changes to 1. With this condition, the only hazard situations encountered are those shown in Figure 3. The first hazard is a static 0 hazard. Where the output was supposed to stay 0 but observes spurious transitions to 1 before settling to final value 0. The other hazard condition

Figure 3: Hazard Values

is a dynamic hazard, where the output was suppose to change from 0 to 1 but observes a number of spurious transitions between 0 and 1 before settling to a final value of 1. Restricting the initial state of the storage elements to 0 helps in two ways: first one does not have to keep track two values

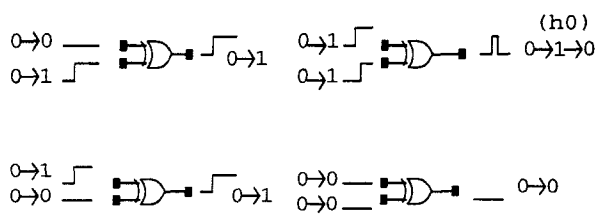

Figure 4: XOR hazard behavior

(initial and final) for each line as in [3], second only two hazard conditions have to be tracked in the network as opposed to four in[3]. This helps in reducing the size of solution space. The hazard generation, hazard filtering, and hazard propagation that could occur in these types of circuits is described below.

5.1 Hazard Generation

Hazards are generated by XOR elements in the circuits. The C-elements do not generate any hazards if the input transitions are hazard free. The only time a hazard can be generated in the circuit is when both the inputs of an XOR make transitions from 0 to 1. In this case, due to unequal delays in the paths of the two inputs, the output may see a

static hazard as shown in Figure 4. This condition constitutes the hazard generation condition in the circuit. At this point one could generate tests in which no XOR in the circuit ever gets 11 as justification values for 0 at the output of XOR. However this may result in no test being generated for a fault, even if one exists with hazards but not resulting into any steady state error. Note that as long as long as the hazard does not cause any steady state error in a test, it can be used as a valid test. Hence the generation of hazard should not stop the search for a test in a circuit.

5.2 Hazard Filtering

The output of a C-element depends only on the steady state values of the inputs if those input values are the same.In other words the C-element output changes only after both the inputs have changed. This behavior of C-element helps in filtering some hazardous signals in the circuit. Input conditions under which a C-element acts as a filter for hazards are shown in Figure 5.

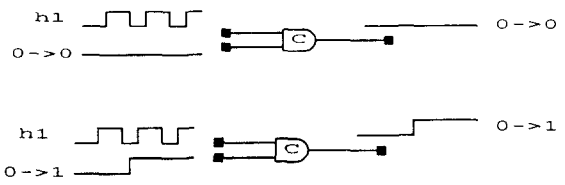

Figure 5: Hazard filtering through C-element

5.3 Hazard Propagation

Another thing one needs to keep track of in critical hazard free test generation is how hazards are propagated through the elements i.e. when the inputs of the elements are hazardous what is the output? Various cases that occur in the circuit are shown in Figure 6. In case of XOR the output waveform is just the XOR of input waveform. However in the case of C-element the output follows the inputs only when both inputs are at the same value otherwise it retains the previous value. So if the steady state value of the inputs is the same then the steady state value of the output is also the same as that of the inputs and the output may be hazardous if the input values are hazardous, But if the steady state value of the inputs is different then the steady state value of the output will depend on the shape of the input waveforms and their timing relationship, which depends upon the circuit delays. The output in this case can not be determined under the unbounded delay assumption. This is the case where a steady state error could occur and the test may get invalidated. In this case the output is defined as B (for Bad).

6 Algebra

The proposed algebra for formalizing the tracking of hazard generation, propagation and filtering is described in the following subsections.

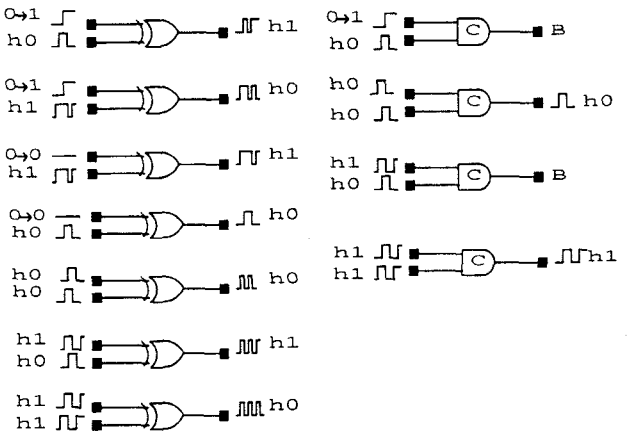

Figure 6: Hazard propagation through elements

6.1 Definitions

The six basic values used in the new algebra are {X, 0, 1, h0, h1,B}. The meaning of these values is described in Table 1. Both good and faulty circuits can assume any of these values. This gives rise to eight composite values a/b where a is the value of a signal in the good circuit and b is the value in the faulty circuit. In this paper we have adopted a convention similar to the one used in the D-algorithm[1] for representing composite values, but in addition we have also tagged symbols with the hazard information. A composite value in this paper made of three parts XYZ where X

Table 1: Basic Values

Val	Meaning
X	Unknown. The value is not yet determined
0	0->0. The value stays 0 after the application of the input vector
1	0->1. The value changes from 0 to 1 after the application of input vector.without any hazards
h0	steady state value is 0. but may have spurious transitions to 1.
h1	The steady state value is 1 but may have spurious transition while changing from 1.
B	steady state error may occur.

Table 2: Composite values

Val		Meaning
Dnn	1/0	Both good and faulty values are hazard free
Dnh	1/h0	Value in the faulty circuit is hazardous
Dhn	h0/1	value is good circuit is hazardous
Dhh	h1/h0	Both values are hazardous
DBnn	0/1	Both values are hazard free
DBnh	0/h1	Value in the faulty circuit is hazardous
DBhn	h0/1	Value is good circuit is hazardous
DBhh	h0/h1	Both values are hazardous

is either D or DB (short form for Dbar) depending on the steady state values of 1/0 or 0/1. Y is h or n depending on whether the value in the good circuit is hazardous or not, and Z is h or n depending upon whether the value in faulty

circuit is hazardous or not. Using this convention the eight composite values obtained are described in Table 2.

One additional basic value B, is used for representing the value which results from incorrect latching of a C-element leading to a steady state error caused by a hazard.

In fact one could actually stop the exploration in the solution space in this direction of the search tree if a B value is seen and then backtrack. In that case this B value will not be required. However it is possible in some tests, a B is generated on only some paths and may even get killed along that path and other paths may have fault propagation to certain outputs. In this case it is better to retain this error value and continue the search in hope that fault propagation along some other path may be possible.

One point to notice is that this incorrect latching could be in the good circuit or the bad circuit. In either case only one value is used for representing all the error cases. This is due to the fact that a bad B value can not be used for error detection as there is no guarantee of good and faulty circuit output to be different in all delay situations. So the only operation that can happen to this value is filtering through a C-element in which case the output is 0. So we do not have to retain different error values for all the cases.

6.2 XOR and C-element operation

The operations of XOR and C-elements over this new set of values are shown in Tables 3 and 4. The interesting entries in the basic behavior part have been shown shaded and represent hazard generation or filtering as described earlier. One important thing to notice from this table is that only two hazard conditions considered in this paper are sufficient to model all the hazard situation that arise in these circuits to the point that hazard values do not result in steady state errors in the C-elements. This observation can be made from the top left portion of the tables which covers only basic logic values.

6.3 Error Detection Criterion

The detection condition in this new algebra is asserted when the any of the faulty values {Dnn, Dhn, Dnh, Dhh, DBnn, DBhn, DBnh, DBhh} appear at the output with no unjustified lines. This extended set for detection means that the presence of a hazardous value on good and/or faulty circuit is acceptable because even though a hazard was generated in the circuit it did not result in any incorrect value being latched in a C-element on the propagation path (otherwise a B would have resulted). Also in the static test environment assumed in this paper, these values will eventually settle to their final steady state value.

7 Implementation and Example

The algebra has been incorporated in a test generation program based on the D-algorithm and has been integrated with the ACT (Asynchronous Circuit Testing) DFT tool[6]. It uses the basic engine of the D-algorithm and uses the enhanced detection conditions described above. An example of test generation is shown in Figure 7. This example is designed specifically to demonstrate some interesting cases of the 6 valued logic. Here a test for P (shown in circle) s-a-0 is generated. The interesting values are shown underlined. First consider the input values of C-element with output P. Here the 1 at line P is justified using 1 and h1 because 1 and 1 values lead to failure in propagation. The use of 1 and h1 to justify 1 at the output of C-element is acceptable because from the table the output of a C-element for these values is 1. The second case is the C-element with M (shown in circle) as its output line. Here a bad value B appears because of 1 and h0 inputs. However the generation of this values is acceptable because it gets killed at the subsequent C-element which evaluates to 0. The third interesting case is at the circuit output Y. Here even though the output value in the good circuit is hazardous the fault is still detected because this hazard does not cause any steady state error

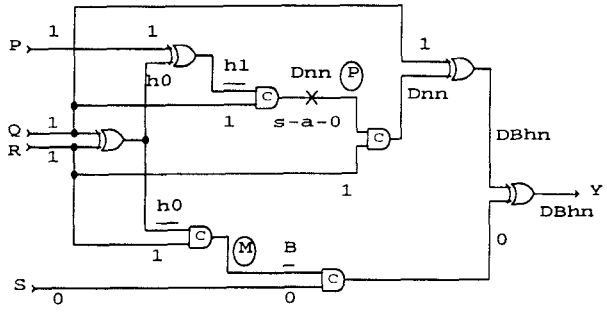

Figure 7: Test generation example

and will cease in the static test environment.

8 Conclusions

A new method has been proposed to generate critical hazard free tests for the control path of an important class of asynchronous circuits called macro-module based self-timed circuits. A new algebra has been developed to automate the test generation process. This algebra needs a much smaller set of values as compared to the approach proposed in[3], hence reduces the size of the solution space and the complexity of test generation. This algebra along with the test modifications proposed for the circuits allows the tests to be generated in a way similar to that for a combinational network. An automatic test pattern generation tool has been developed using this algebra under the framework of the D-algorithm. This tool has been integrated with the ACT[6] DFT tool.

9 References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[2] K. V. Berkel, R. Burgess, J. Kessels, M. Roncken, F. Schalij, and A. Peeters. Asynchronous Circuits for Low Power: A DCC Error Corrector. IEEE Design and Test of Computers, vol 11, Summer 1994.

[3] M. A. Breuer and R.L. Harrison. Procedures for eliminating static and dynamic hazards in test generation, IEEE Transactions on Computers, c-23:1069--1078, 1974.

[4] E. Brunvand. The NSR Processor, In Proceedings of the 26th International Conference on System Sciences, Maui, Hawaii, January 1993.

[5] E. Brunvand. *Translating Concurrent Communicating Programs into Asynchronous Circuits*, PhD thesis, Carnegie Mellon University, 1991.

[6] A. Khoche. *Testing macro-module based self-timed circuits*, PhD thesis, University of Utah, 1996.

[7] A. J. Martin. Compiling Communicating Processes into Delay-insensitive Circuits, Distributed Computing, 1(3), 1986.

[8] A. J. Martin, S. M. Burns, T. K. Lee, D. Borkovic, and P. J. Hazewindus. The Design of an Asynchronous Microprocessor, Advanced Research in VLSI. MIT Press, 1990.

[9] C. L. Seitz. System Timing, In Mead and Coway, *Introduction to VLSI Systems, chapter 7*, Addison-Wesley, 1980.

[10] Ivan Sutherland. Micropipelines, CACM, 32(6), 1989.

[11] R. Sproull, I. E. Sutherland, and C. E. Molnar. Counterflow Pipeline Processor Architecture, IEEE Design and Test of Computers, Fall 1994.

[12] N. Paver. *The Design and Implementation of an Asynchronous Microprocessor*, PhD Thesis, University of Manchester, UK, 1994.

Table 3: XOR Table

	X	0	1	h0	h1	Dnn	DBnn	Dhh	DBhh	Dhn	DBhn	DBnh	Dnh	B
X	X	X	X	X	X	X	X	X	X	X	X	X	X	B
0	X	0	1	h0	h1	Dnn	DBnn	Dhh	DBhh	Dhn	DBhn	DBnh	Dnh	B
1	X	1		h1	h0	DBhn	Dnh	DBhh	Dhh	DBhn	DBhh	Dnh	DBhh	B
h0	X	h0	h1	h0	h1	Dhh	DBhh	Dhh	DBhh	Dhh	DBhh	DBhh	Dhh	B
h1	X	h1	h0	h1	h0	DBhh	Dhh	DBhh	Dhh	DBhh	Dhh	Dhh	Dhh	B
Dnn	X	Dnn	DBhn	Dhh	DBhh	h0	1	h0	h1	h0	h1	h1	h1	B
DBnn	X	DBnn	Dnh	DBhh	Dhh	1	h0	h1	h0	h1	h0	h0	h0	B
Dhh	X	Dhh	DBhh	Dhh	DBhh	h0	h1	h0	h1	h0	h1	h1	h0	B
DBhh	X	DBhh	Dhh	DBhh	Dhh	h1	h0	h1	h0	h1	h0	h0	h1	B
Dhn	X	Dhn	DBhn	Dhh	DBhh	h0	h1	h0	h1	h0	h0	h1	h1	B
DBhn	X	DBhn	Dhh	DBhh	Dhh	h1	h0	h1	h0	h1	h0	h0	h1	B
DBnh	X	DBnh	Dnh	DBhh	Dhh	h1	h0	h1	h0	h1	h0	h0	h1	B
Dnh	X	Dnh	DBhh	Dhh	DBhh	h0	h1	h0	h1	h0	h1	h1	h0	B
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B

Table 4: C-element Table

	X	0	1	h0	h1	Dnn	DBnn	Dhh	DBhh	Dhn	DBhn	DBnh	Dnh	B
X	X	0	X	X	X	X	X	X	X	X	X	X	X	B
0	0	0	0			0	0	0	0	0	0	0	0	0
1	X	0	1	B		Dnn	DBnn	B	B	Dnn	B	DBnn	B	B
h0	X		B	h0	B	B	B	B	B	B	B	B	B	B
h1	X			B	h1	Dnn	DBnn	B	B	Dhn	B	Dnh	B	B
Dnn	X	0	Dnn	B	Dnn	Dnn	0	Dnn	B	Dnn	B	0	Dnn	B
DBnn	X	0	DBnn	B	DBnn	0	DBnn	B	DBnn	0	DBnn	DBnn	B	B
Dhh	X	0	B	B	B	Dnn	B	Dhh	B	Dhn	B	B	Dnh	B
DBhh	X	0	B	B	B	B	DBnn	B	DBhh	B	DBhn	DBnh	B	B
Dhn	X	0	Dnn	B	Dhn	Dnn	0	Dhn	B	DBhn	B	0	Dnn	B
DBhn	X	0	B	B	B	B	DBnn	B	DBhn	B	DBhn	DBnn	B	B
DBnh	X	0	DBnn	B	DBnh	0	DBnn	B	DBnh	0	DBnn	DBnh	B	B
Dnh	X	0	B	B	B	Dnn	B	Dnh	B	Dnn	B	B	Dnh	B
B	X	0	B	B	B	B	B	B	B	B	B	B	B	B