Proceedings of the 10th International Conference on Applied Informatics
Eger, Hungary, January 30–February 1, 2017. pp. 203–210
doi: 10.14794/ICAI.10.2017.203

Simulation Environment for Investigation of Delay-Insensitivity of Data Flow Structure Asynchronous Networks and Systems

Attila Nagy, Péter Keresztes

Széchenyi István University, Department of Mechatronics, Győr, Hungary
anagy@sze.hu, keresztp@sze.hu

Abstract

Clocking is one of the most significant problems of VLSI system design. It is not easy to give a general definition of delay-insensitivity, but it is even more difficult to verify it for a given digital system. The simulation method proposed by the authors makes it possible to follow the classical bottom-up design method from switch level to register-transfer level.

The first step of the proposed designing process is to define a structure consisting of delta-delay DI components in VHDL. The following step of the design process is inserting pulse controlled sample-and-hold type switches into the delta delay architecture. These components are virtual, physically not realised models, and they are referred to shortly as S&Hs in the paper. If all gates or cells and their interconnections were represented by the S&Hs, and all possible sequences including simultaneous activations executed, a simulation with positive results could be considered a verification of delay-insensitivity. The paper presents the special VHDL-FLI simulation environment developed, the design and simulation process, and shows several interesting problems in which the author's method played an important role.

Keywords: Delay-insensitive (DI) asynchronous networks, Simulation, VLSI design

MSC: AMS classification numbers

1. Introduction

The special field which the paper discusses appeared in the middle of the last century, and it was referred to as the speed-independence of logic circuits. Delay of logic gates dominated over the delay of wires and interconnections at that time. The gate delays were the main source of many disadvantages (static and dynamic hazards, critical race situations) of the classical asynchronous circuits.

Müller investigated autonomous asynchronous circuits, determined the conditions of speed-independence, and created his famous C circuit, which is a simple classical asynchronous network with logical hysteresis, and later became the key component of the new DI networks and systems. Also Müller introduced the design method of DI combinational circuits, which is called DIMS. (Delay-Insensitive Minterm Synthesis). Müller-C is not only the most important components of DIMS, but of a later version of DI combinational networks, NCL (Null Convention Logic) too. Another approach was the research of DI communication protocols between interconnected black-box models. Verhoeff gave a general solution for DI binary codes, van Berkel combined them with his four-phase asynchronous communication protocol.

The conditions of delay-insensitivity were the subject of many works. Udding [1], Dill [2] and others used several formal methods as trace theory and theory of communicating sequential processes.

In the first decade of the new century, finally a commonly accepted version of DI modules took shape, with the following features:

- Active output and passive input for data represented with delay insensitive code
- One of the two handshake signals, signal request is hidden in data.

There is an analogy between this version of DI networks and the behaviour of static data flow graphs introduced by Dennis [3, 4].

Martin [5] investigated the limitations of delay-insensitivity. From his work, it became clear that the number of DI networks is absolutely limited. A much larger number of networks designed carefully can be considered so called quasi delay-insensitive.

2. Simulation with S&H switches

In the last decades some simulation methods were developed for checking the delay issues related to the asynchronous network. None of them is perfect, however. Some of them are too simple or too conservative [6], while others require huge computation effort.

The method presented in this paper can support the classical bottom-up design method of the quasi delay-insensitive (QDI) networks. Among several advantages of these circuits, the most important one is that the DI circuits can work properly, independently from the delays of wires and gates. Apart from some very simple circuits it is not possible ensuring the delay-insensitivity for each gate and wires. In order to be able to verify the delay-insensitivity of certain wires or gates, the delay elements of these components can be modeled by simple S&H switches, which let through the signal from their input to their output. The replaced elements can be considered delay-insensitive: 1) if for each possible switching sequence the circuit works properly and 2) a changed value on the input of a S&H switch must not

change back before it is passed by the switch. For not very complicated circuits, the S&H switches can be actuated manually, but for more sophisticated circuits, the checking process requires an automatism.

In the next subsections, two examples are shown which demonstrate how the proposed method works in case of manually activated S&H switches. Section 3 briefly presents the automatically activated S&H switch implementation using VHDL models of the DI circuits and the Foreign Language Interface (FLI) of the Modelsim.

2.1. A simple example for application

Truth table of a very simple QDI unit is given in Table 1 The function is a code conversion between two DI codes, namely a 2-of-3 and a code represented by two dual rail variables. The wires of input x are *X2*,*X1*,*X0*, and the wires of output variables *y1* and *y2* are *Y11*, *Y10*, *Y01*, *Y00*. The signals *ax* and *ay* are the acknowledge signals of the input and the output respectively.

x	X2	X1	X0	y1	y0	Y11	Y10	Y01	Y00
0	L	H	H	0	0	L	H	L	H
1	H	L	H	0	1	L	H	H	L
2	H	H	L	1	0	H	L	L	H

Table 1: Truth table of 2-of-3 – two variable dual-rail code converter

The behavioural VHDL description is shown in the first point of the APPENDIX. The signal assignment statements of this description are free of so called after clauses, so it can be considered a pure delta-delay model. This behaviour mirrors the causality between signal transitions. If a transition is a direct consequence of another, the simulation shows a single DELTA delay between them. If the causal chain is longer, the multiplied DELTA delay shows the length of the chain.

Figure 1 shows the application of our simulation method for the verification of delay-insensitivity of BDM (Behavioural Delta Model), i.e. a simulation which has shown that the correctness of the communication remains unaffected by the delays of buses to and from a transmitter and a receiver. Note several properties of the VHDL abstract receiver model. If *ax* is low, all the three push outputs of the S&H switches of it goes high, but if you activate two pass inputs, the third push output goes low.

The following step of a bottom-up designing process is to build up the gate level architecture of the code converter. On Figure 2 the structure of gate level implementation is shown. The triplet of the three-input Müller-C units and the 3 input OR constitute a completion detector for the 2-of-3 code, which drives the acknowledge signal *ax*. At the same time the C-triplet is an asynchronous register, the outputs of which are connected with OR gates of output part.

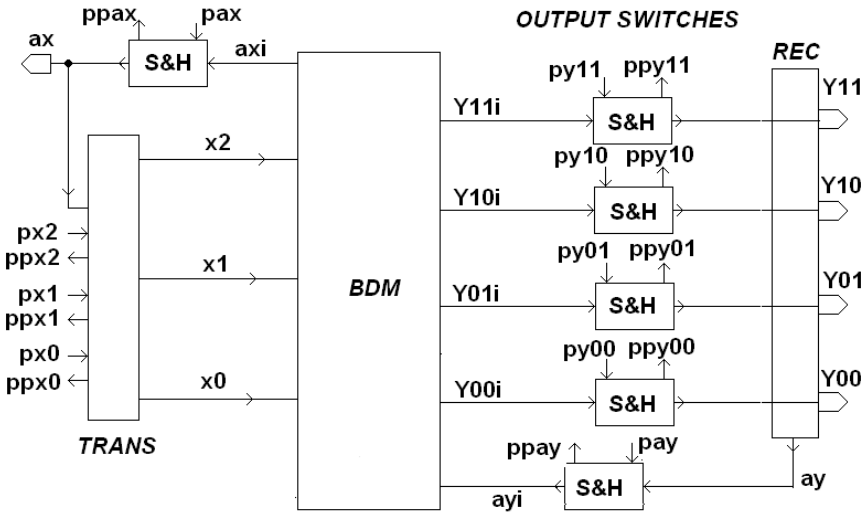

Figure 1: Simulation model for verification of delay-insensitivity of BDM inserted in a real delay communication environment. (ppx_i : push px_i button; px_i : pass signal x_i)

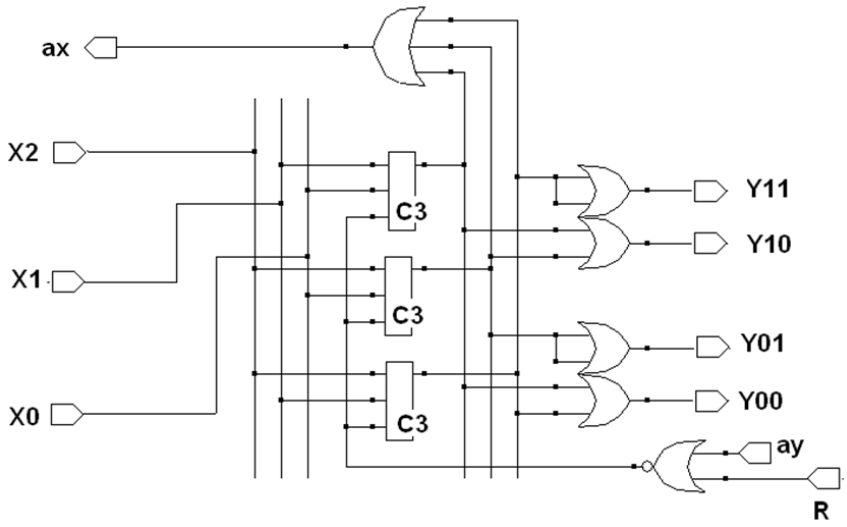

Figure 2: Gate level structure of code converter with 3 input Müller-C components

The simulation of the gate level structure completed with S&H switches (Figure 3) has shown that the gate level implementation is not delay-insensitive, it is

quasi delay-insensitive, if the branch of input *X1* to the upper and lower C units can be considered an isochron branch. The following pass-sequence leads to a communication error: *pX0*;*pX1*;*pX1f*;*pX1s*;*pay*;*pX1*;*pX0*;*pX1f*;*pay*;*pX2*;

Here *pX1f* is the pass input of the S&H representing the fast wire, *pX2s* is the pass input of the S&H representing the slow wire of the branch of *X1*.

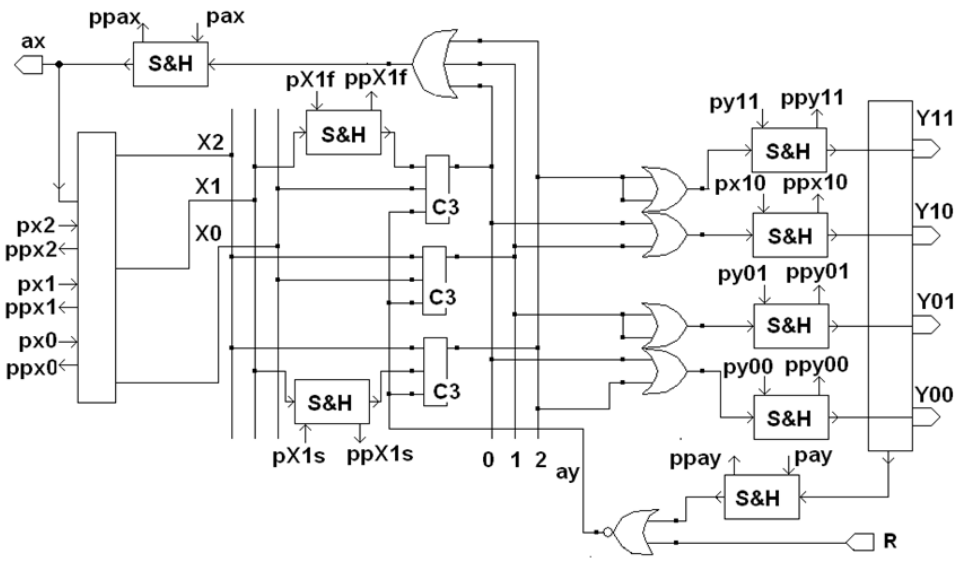

Figure 3: Example for simulation of slow wires

2.2. Verification of a new MERGE unit for QDI ring applications

The simulation using the new delay modeling method has made it possible to find a new QDI MERGE circuit, the application of which makes a QDI multiplexer, used as an input stage of a ring, redundant. QDI ring plays an important role in pipelines executing iterative and repetitive calculations. The critical feature of the ring with a conventional MERGE unit can be that the feedback should be faster than the initial state setting communication, so there is a communication overlap on the inputs of the register-level scheme given in Figure 4. Evaluating the simulation results has shown that the acknowledge signal *asa* has to be faster, than the enable signal *esab*. This demand can be satisfied in a careful layout design, independently from the other parts of the ring. So the delta-model of this carefully designed MERGE unit is very simple: An additional DELTA-delay has to be inserted into the structure, consisting of DELTA models of components. ($esab <= asa$;)

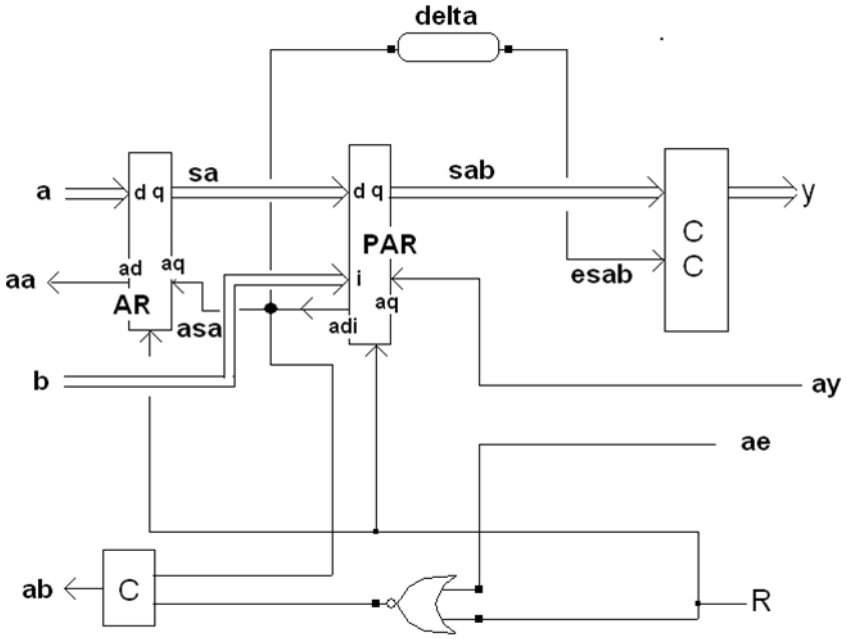

Figure 4: A new MERGE unit for QDI ring applications. AR : Asynchronous Register, PAR : Presettable Asynchronous Register, CC : Column of C units

3. Automatic simulation of VHDL models

Because of the rapidly increasing number of the possible switching sequences, simulation with manually activated switches strongly limits the number of S&H switches. In order to be able to test the delay-insensitivity of more complicated circuits, the authors developed a simulation environment for automatic switching. Figure 5 shows the internal states of the simulation. In spite of the simple concept using a recursive backtracking program, the realization is quite complicated. The system goes to the next state along the solid lines when a S&H switch changes its state. The dashed lines represent the restarting of the simulation from a certain moment. At the beginning of the simulation in *Idle* state when the input of a S&H switch changes, the FLI program terminates the simulation and saves the full state of the simulation in order to be able to restart the simulation from this moment several times. In *Collect* state the program collects all events whose switching sequences have to be permuted. After the *Collect* phase the simulation is restored to the *Break* moment and the outputs of the switches are *Driven* by the FLI program.

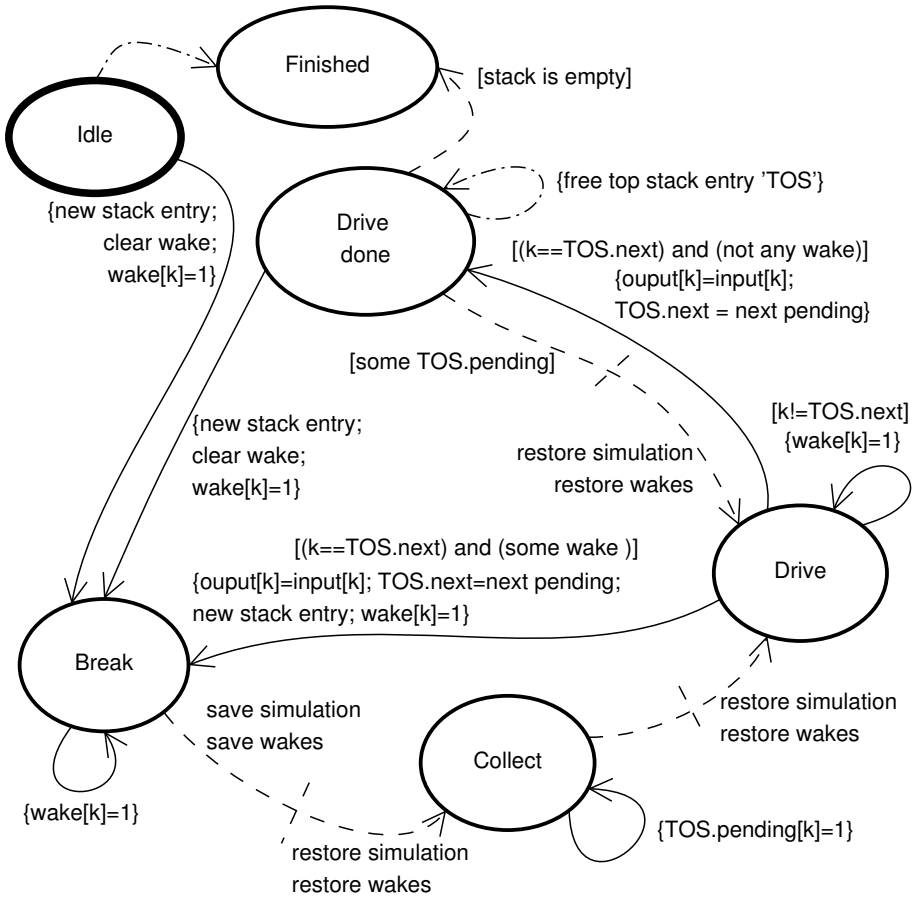

Figure 5: Internal states of the FLI program

4. Appendix

4.1. VHDL description of the simple example

```
entity DI_CODE_CONVERTER is
port( X2, X1, X0 : in bit;
    Y11, Y10, Y01, Y00: inout bit;
    ax : out bit;
    ay : in bit);
end;
architecture BDM of DI_CODE_CONVERTER is
begin
    process(X2, X1, X0, ay)
```

```
    begin
    if ay = '0' and X2 = '0' and X1 = '1' and X0 = '1' then
        Y11 <= '0'; Y10 <= '1';Y01 <= '0'; Y00 <= '1'; ax <= '1';
    elsif ay = '0' and X2 = '1' and X1 = '0' and X0 = '1' then
        Y11 <= '0'; Y10 <= '1'; Y01 <= '1'; Y00 <= '0'; ax <= '1';
    elsif ay = '0' and X2 = '1' and X1 = '1' and X0 = '0' then
        Y11 <= '1'; Y10 <= '0'; Y01 <= '0'; Y00 <= '1'; ax <= '1';
    elsif ay = '1' and X2 = '0' and X1 = '0' and X0 = '0' then
        Y11 <= '0';Y10 <= '0'; Y01 <= '0'; Y00 <= '0';ax <= '0';
    end if;
    end process;
end BDM;
```

4.2. VHDL statements for S&H switches

Let the ports of a switch be *i*, (input), *o*, (output), *pi*, (pass input) and *ppi* (push pass input). Two signal assignment statement can define its operation, as follows

```
o <= i when pi = '1' else o;
ppi <='1' when o /= i else '0';
```

References

[1] J.T. Udding, A formal model for defining and classifying delay-insensitive circuits and systems. *Distributed Computing,* 1986, 1:197-204.

[2] D.L. Dill, Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits. *The MIT Press,* Cambridge, Mass.,1988. An ACM Distinguished Dissertation 1988.

[3] J.B. Dennis, The Evolution of State Data Flow Architecture. *Advanced Topics of Data Flow Computing. J.L. Gaudiot and L.Bic, eds., Prentice Hall,* 1991

[4] P.Keresztes, L.T.Kóczy, A.Nagy, G.Rózsa, Training Elecctrical Engineers on Asynchronous Logic Circuits Based on Constant Weight Codes. *Proceeding of IEEE Africon 2011*, Livingstone, Zambia, no : NF002631, DOI: 10.1109/AFRCON.2011.6072041, pp. 1-7

[5] A. Martin., The limitation to delay-insensitivity in asynchronous circuits. In *Proceedings of the Conference an Advanced Research in VLSI,* April 1990.

[6] J.A. Brzozowski, Delay-insensitivity and ternary simulation. *Theoretical Computer Science,* 2000, 245:3-25