Modeling of Real Bistables in VHDL

Acosta, A.J.[1], Barriga, A.[1], Valencia, M.[2], Bellido, M.J.[2] and Huertas J.L.[1]

Dept. of Design of Analog Circuits. Centro Nacional de Microelectrónica, Edificio CICA, Avda. Reina Mercedes s/n, 41012-Sevilla (Spain)

[1] also with the Dept. de Electrónica y Electromagnetismo, Universidad de Sevilla.
[2] also with the Dept. de Tecnología Electrónica, Universidad de Sevilla.

Abstract

A complete VHDL model of bistables including their metastable operation is presented. An RS-NAND latch has been modelled as a basic structure, orienting its implementation towards its inclusion in a cell library. Two applications are included: description of a more complex latch (D-type) and description of a circuit containing three latches where metastable signals are propagated. Simulation results show that the presented model provides very realistic information about the device behavior, which until now had to be obtained through electric simulation.

1: Introduction

Precise modelling of real digital systems or components through VHDL is an interesting task if we wish, on one hand, to maintain the inherent advantages of high level modelling and simulation, and on the other, adequately describe these circuits. The simplicity of describing digital circuits is seriously affected when their performance is not ideal, or simply when, for specific applications, they must be modelled most precisely. In many applications, the logic abstraction level for circuit descriptions requires a precise behavior model of its components. In our case, we refer to memory elements operating in metastability.

This communication presents real descriptions of digital memory elements like RS-type asynchronous latches, and their use to describe D-type level-sensitive bistables. These elements are indispensable to perform sequential systems. The VHDL description found in literature of this type of elements only takes into account normal behavior, or in some cases, has implemented routines to detect abnormal behavior [1,2]; we encountered no reference in literature regarding the behavior modelling of memory elements when these operate abnormally or in metastability. The importance of this model lies not only in the device itself, but also in the pernicious influence that this circuit may have on its environment, which can also be modelled. The objective of this communication is the implementation of VHDL routines for metastable behavior modelling.

This paper is structured as follows: Section 2 is a brief review of the aspects related to metastability in bistables; Section 3 includes VHDL models of generated latches; and Section 4 presents and discusses two applications of the model generated. Appendix A presents graphic and numerical results of simulations made with these models.

2: Metastable operation in bistables

When a bistable is triggered by signals violating its time restrictions (setup time, hold time,...), it operates at a point of unstable equilibrium called metastable state [3-5]. These excitations, also called marginal triggering, drive the bistable from one stable state to the metastable state, to resolve one of the logical stable states after a time which depends on the bistable and the marginal triggering itself. The metastable operation may be modelled assuming that the logic value of the bistable output is indefinite for an unbounded interval of time, after a marginal triggering. Consequently, metastability is a potential source of errors, which is why we consider most interesting the contemplation of simulation processes of this phenomenon. At a functional level, the main problems are, on one hand, determine which marginal excitations lead to metastable operation of the bistable, and on the other, for a specific marginal triggering, what is the time that the output maintain an indefinite value and what is the final state reached.

The approach that we consider most adequate is to model the basic bistable (that present basic metastable operation), and from this, model more complex bistables. For this reason, initially we will only consider the case of an RS-type latch constructed with crossed NAND gates (Figure 1a). The case of an RS latch constructed

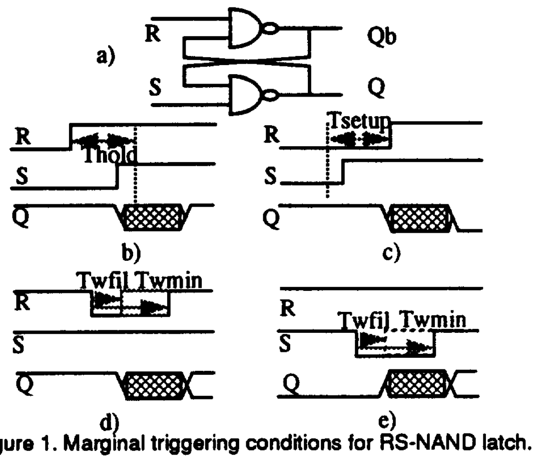


Figure 1. Marginal triggering conditions for RS-NAND latch.

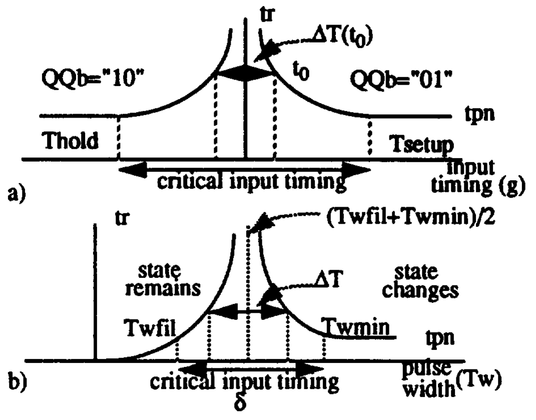


Figure 2. Critical input timing for the RS-NAND latch: a) simultaneous transition of R and S; b) runt pulses.

with NOR gates is equivalent and is not presented for brevity in the communication.

The excitations causing metastability in the RS-NAND latch are:

a) Violation of the hold time (Thold): The time interval (g) between the rise of the R signal and the rise of the S signal is less than the hold time (Figure 1b).

b) Violation of the setup time (Tsetup): The time interval (g) between the rise of the S signal and the rise of the R signal is less than the setup time (Figure 1c).

c) Violation of the minimum pulse width: A narrow pulse (width less than Twfil) is filtered by the latch due to its inertial effect, and a pulse wider than Twmin makes the latch operate normally. If its width is intermediate (runt pulse) it may cause metastability, when the state imposed by the pulse does not coincide with the stored state (Figures 1d and 1e).

The parameters cited (Tsetup, Thold, Twfil, Twmin) depend on the bistable implementation.

Numerous work for different types of bistables and technologies have been presented to adequately model metastable operation of bistables. A model amply used, both theoretically [5] and experimentally [3] is the so-called exponential model, which provides a reasonably simple explanation of the metastability of any bistable. This model establishes the existence of an exponential relation between the minimum residence time in metastable state of a bistable (tr) and the time interval ($\Delta T(tr)$) in the excitation space within the critical input timing window. Algebraically [5]:

$$\Delta T(tr) = \delta e^{tpn/\tau} e^{-tr/\tau} \qquad tr>tpn \qquad (1)$$

where δ is the width of the critical input timing window, tpn the normal propagation time outside of the critical input timing window, and τ is a parameter whose value depends on each bistable. Usually, RS-NAND latches are symmetric. Then, a graphic representation of the model is presented in Figure 2a, for those cases of violation of setup and hold, and in Figure 2b, for the case of metastability for a runt pulse.

In the case of near or simultaneous transition of the R and S signals (Figure 2a) we represent on the x-axis the time interval "g" existing between the transitions of both signals, and on the y-axis the bistable resolution time "tr". For this type of marginal triggering, $\Delta T(tr) = 2g(tr)$ and δ=2*Tsetup, since Tsetup=Thold for symmetric bistables. With this model, we obtain from expression (1) the following expression for entering metastability due to the violation of setup or of hold:

$$tr = tpn + \tau \log \left(\frac{Tsetup}{g}\right) \qquad (2)$$

Thus, the bistable presents normal propagation time (tpn) outside of the critical input timing region and when entering metastability, the normal propagation time increases by a quantity that depends on the τ parameter, on the time interval g, and on the time parameter Tsetup.

For the case of excitation by a runt pulse, the critical input timing window is the shown in Figure 2b. Upon assuming symmetric bistables, the window is symmetric for tr > tpn. This is equivalent to assuming that all pulses of width Twfil<Tw<Twmin provoke metastability. The center of the window coincides with the width of pulse T=(Twfil + Twmin)/2. In the limit where Tw=Twmin, tr equals tpn, and we can determine from Figure 2b that

$$\Delta T(tpn) = \delta = Twmin - Twfil$$

and from (1) we deduce that:

$$tr = tpn + \tau \log \left(\frac{Twmin - Twfil}{2\left| Tw - \left(\frac{Twfil + Twmin}{2}\right) \right|} \right) \qquad (3)$$

Eqs. (2) and (3) are the core of the proposed model.

3: Modelling in VHDL

Our approximation consists of declaring the bistable as a component, to which arguments are passed as a group of parameters, depending on the technology and type of bistables. These parameters include those relating to metastable operation (τ). When this element is invoked from a VDHL file, a sequential algorithm is executed, in such a way that it generates the output waveforms from changes in the input signals.

With the strategy presented, the user must know the τ value appropriate for the type of bistable used. This may be inconvenient since this data does not usually appear in the library catalogues of bistables. However, this parameter can be measured experimentally [3] or by electric simulation [4].

First, let us consider a simple model of this type of latch (Figure 3). Basically, the file consists of an entity declaration and an architectural body, in which all possible combinations of the input and output corresponding to each of them are covered by an if-else clause.

```
entity latchrs is
 generic(tpn: time); --tpn: normal propagation delay
 port( R, S: in bit; QQb: out bit_vector (1 downto 0));
end latchrs;

architecture logical of latchrs is

begin
  process
  variable RS: bit_vector (1 downto 0);
            begin
            wait on R,S;
            RS:=R&S;
            if RS="10" then
                 QQb<="10" after tpn;
            elsif RS="01" then
                 QQb<="01" after tpn;
            elsif RS="00" then
                 QQb<="11" after tpn;
            end if;
  end process;
end logical;
```

Figure 3. VHDL description of a simple model of an RS latch

Some routines to detect violations of the time parameters (setup, hold, or pulse width) appear in [1,2], implemented as procedures that run whenever there is a change in the input signals. These verification procedures run assert-report statements in the case of time violation and are included in our models. However, these concurrent instructions only detect the existence of metastable behavior and do not model the behavior of the device under these conditions. Figure 4 presents the code that reflects such behavior. The model is obtained through a process which distinguishes five parts: one for normal behavior, and the other four describe the metastable behavior for the four causes mentioned: violation of hold, setup, S-runt pulse and R-runt pulse.

```
library WORK;
use WORK.functions.all;

entity latchrsnand is
 generic( tpn,tsetup,tau,Twfil,Twmin: time);
 port( R, S: in MVL; QQb: out MVL_vector (1 downto 0));
end latchrsnand;

architecture logical of latchrsnand is
   type MVL is ('X','0','1');
    type MVL_vector is array (NATURAL range <>) of MVL;
signal o,flagR,flagS:MVL:='0';
begin
 process
 variable aux:time;
 variable RS: MVL_vector (1 downto 0);
        begin
        wait on R,S;
        RS:=R&S; --variable introducted for simplicity
        if RS="10" then
               if (not S'stable) then
                      flagS<='1';
               end if;
               o<='1';
               QQb<="10" after tpn;
        elsif RS="01" then
               if (not R'stable) then
                      flagR<='1';
               end if;
               o<='0';
               QQb<="01" after tpn;
        elsif RS="00" then
               flagR<='0'; flagS<='0';
               o<='1';
               QQb<="11" after tpn;
        elsif (not R'stable and S'last_event<Tsetup) then
*In this space, code for setup violation is included (Figure 5)*
        elsif (not S'stable and R'last_event<Tsetup) then
*In this space, code for hold violation is included;*
        elsif (not S'stable and flagS='1' and flagS'las-
t_event<Twmin and o'last_value='0') then
*In this space, code for S-runt violation is included (Figure 6)*
        elsif (not R'stable and flagR='1' and flagR'las-
t_event<Twmin and o'last_value='1') then
*In this space, code for R-runt violation is included;*
        else
               flagR<='0'; flagS<='0';
        end if;
 end process;
end logical;
```

Figure 4.Proposed VHDL description of the RS-NAND latch.

The first part of the description, called normal behavior, coincides with the basic model of an RS bistable without metastability. The signals "flagS", "flagR" and "o" are introduced as auxiliary signals that will be used later to describe the metastable behavior. The signal "o" stores the value of the output signal "Q", which, since it is declared in out mode cannot be read. The signals flagS and flagR are used to determine at which instant the signals S and R experienced their last transition. These data will be used to measure the width of pulses R and S.

The detection of metastability is evaluated for each of the four last if-else clauses, and the corresponding code lines run if conditions are met. Appendix A presents simulation results of the implemented model.

Another modification included is the declaration of MVL (Multi-Valued Logic) signals in addition to logic levels. The definition of this type of signals (Figure 4) includes a third level, 'X', that we will use indistinctly to indicate both indeterminate signals and metastable behavior. This definition is necessary to distinguish normal operation from metastable operation with two particular intentions:

a)Reflect the inherent logic indetermination of metastable behavior.

b)Model the transmission of the metastable state from one bistable to another or to any other device that is in contact with the latch.

3.1: Setup violation

```
assert S'last_event<tsetup report "setup violation"
severity note;
   flagR<='0'; flagS<='0';
   if S'last_event<=0.001*tsetup then
           aux:=0.001*tsetup;
   else
           aux:=S'last_event;
   end if;
   QQb<="XX";
   wait until R='0' or S='0' for tpn+tau*ln(tsetup,aux);
   if R='0' then
           o<='0';
           QQb<="01" after tpn;
   elsif S='0' then
           o<='1';
           QQb<="10" after tpn;
   else
           o<='0';
           QQb<="01";
   end if;
```

Figure 5. VHDL code for setup violation.

Basically, what is evaluated for the detection of metastability provoked by setup violation is the separation existing between the rise of the R signal (at that instant) and the last rise of the S signal. If this separation is less than Tsetup, metastability is entered. The following is an explanation of the code:

a) Line 2 resets the value of the signals flagR and flag S, indicating that there is no possibility of runt pulse. This implies that we consider that there is only one simultaneous cause of metastability.

b) The following if-else clause avoids exact simultaneous transition between R and S (g=0). In this case, according to (1) the resolution time would be infinite, and should be avoided because it is not realistic, since probabilistically there is always some cause that makes the bistable switch towards one state or another in a finite time. The proposed solution deviates the center of the marginal window a thousandth part of the setup time, obtaining finite resolution time. The auxiliary variable "aux" stores the value of g.

c) Outputs Q and Qb are loaded with the indefinite value X after a zero settling time when the marginal triggering occurs, and remain in this state during the time: *tpn +tau*ln(tsetup/aux)*; this is the effective time interval of metastability according to (2). The function ln(a,b)=ln(a/b) should be implemented to this effect (see package "functions", Figure 7), or be available in a mathematical library.

d) The wait clause presents two break conditions, (R='0' or S='0'), that indicate, if there is a change in the R or S signals during metastability, new final states are reached (QQb="01" if the reset signal falls, or QQb="10" if the set signal falls). If the metastable state ends naturally, the final state reached is QQb="01", since the transition provoking metastability was RS: 00 -> (01) -> 11. This implies that we find ourselves on the right-hand side of the window in Figure 2a.

3.2: Hold violation

Basically, the description equals that of setup violation, with the exception that the roles of the R and S signals are exchanged here. If the input signals do not change, once metastability ends the final state is the opposite of that defined in the case of setup violation: QQb="10", since the transition provoking metastability was RS: 00 -> (10) -> 11. This is the same as stating that we find ourselves on the left-hand side of the window in Figure 2a. Thus, for symmetry the hold time is identical to that of setup, for which Thold does not appear as a parameter, and Tsetup is used instead.

3.3: S-runt pulse violation

The detection of metastability provoked by runt pulse is somewhat more complicated. To detect the presence of a pulse and measure its width, an auxiliary signal (flagS or flagR) is necessary to identify the last signal transition. At the same time, the state of the bistable output must be known, since only a pulse may provoke metastability if the bistable state tries to change by pulse action. Metastability is only provoked by pulses whose intermediate widths are between Twfil and Twmin.

Figure 6 presents the VHDL code corresponding to the violation of S-pulse runt. The following is an explanation of the code:

a) The first if-else clause is introduced to filter pulses of width lower than Twfil, and the final state reached is exactly the previous one stored (QQb="01").

b) Concerning intermediate pulses, let us consider two parts of the algorithm, according to pulse width --more or less than (Twfil+Twmin)/2. If (Twfil+Twmin)/2 < Tw <Twmin, the pulse is wide enough to change the state, even though it causes metastability (first part of the algorithm). If Twfil<Tw<(Twfil+Twmin)/2, the pulse is narrow enough to not provoke a change in state, even though it causes metastability (second part of the algorithm). These two cases are included in the main if-else clause.

c) Since we have considered the point in time of infinite resolution, once again due to symmetry, at the point

```
assert (not S'stable and flagS='1' and flagS'last_event
<Twmin and o'last_value='0') report "S-runt violation" se-
verity note;
  if flagS'last_event<Twfil then
            QQb<= "01"; flagR<='0'; flagS<='0';
  elsif flagS'last_event>=0.5*(Twfil+Twmin) then
      if flagS'last_event<=0.5005*(Twfil+Twmin) then
            aux:=0.0005*(Twfil+Twmin);
      else
            aux:=flagS'last_event-0.5*(Twfil+Twmin);
      end if;
      QQb<="XX";
      wait until R='0' or S='0' for tpn+tau*ln(0.5*(Twmin-
Twfil),aux);
      if R='0' then
            o<='0'; QQb<="01" after tpn;
      elsif S='0' then
            o<='1'; QQb<="10" after tpn;
      else
            o<='1'; QQb<="10";
      end if;
  else
      if flagS'last_event>=0.4995*(Twfil+Twmin) then
            aux:=0.0005*(Twfil+Twmin);
      else
            aux:=0.5*(Twfil+Twmin)-flagS'last_event;
      end if;
      QQb<="XX";
      wait until R='0' or S='0' for tpn+tau*ln(0.5*(Twmin-
Twfil),aux);
      if R='0' then
            o<='0'; QQb<="01" after tpn;
      elsif S='0' then
            o<='1'; QQb<="10" after tpn;
      else
            o<='0'; QQb<="01";
      end if;
  end if;
  flagR<='0'; flagS<='0';
```

Figure 6. VHDL code for S-runt violation.

(Twfil+Twmin)/2, pulses of this exact width are "deviated" in a thousandth of this value, so that the resolution time is finite. The auxiliary variable "aux" stores the value of the time interval, according to (3).

d) Outputs Q and Qb are loaded with the indefinite value X after a zero settling time when the marginal triggering occurs, and remain in this state during the time: *tpn +tau*ln(tsetup/aux)*; this is the effective time interval of metastability according to (3).

e) The wait clause presents two break conditions, (R='0' or S='0') indicating, if there is a change in the R or S signals during metastability, new final states are reached (QQb="01" if the reset signal falls, or QQb="10" if the set signal falls). If the metastable state ends naturally, the final state reached is QQb="10", if pulse width is Twfil<Tw<(Twfil+Twmin)/2. This implies that we find ourselves on the right-hand side of the window in Figure 2b. Otherwise, if pulse width is (Twfil+Twmin)/2 < Tw <Twmin, the state remains (QQb="01") i.e. we find ourselves on the left-hand side of the window in Figure 2b.

3.4: R-runt pulse violation

The case of R-runt pulse violation is very similar to that developed for the S-runt pulse violation. Only the final state and the metastability detection conditions change. Also, the R and S signals are interchanged. The parameters (Twmin and Twfil) are the same since the bistable is considered symmetric. The final state reached is QQb="01", if the pulse width is (Twfil+Twmin)/2<Tw<Twmin. If the pulse width is Twfil<Tw<(Twfil+Twmin)/2, the final state is QQb="10", that is, the state does not change. If the pulse width is Twfil>Tw, the pulse is filtered and the final state is QQb="10".

3.5: Package

```
package body functions is
function ln(T,J: TIME) return REAL is
variable P:INTEGER:=1;
variable result:REAL:=0.0;
variable I,Q: REAL:=1.0;
begin
          Q:=1.0-real(J/ns)/real(T/ns);
          loop
               if (1.0/I)*(Q**P) < 0.01*result then
                     exit;
               end if;
               result := (1.0/I)*(Q**P) + result;
               I:=I+1.0; P:=P+1;
          end loop;
          return result;
end ln;
end functions;
```

Figure 7. Package "functions".

The package "functions" (Figure 7) contains an algorithm capable of calculating the natural logarithm of the quotient of two numbers, with less than 1% error. The function ln(a,b) has two TIME parameters as arguments and returns a REAL value (log(a/b)). Implementation of this algorithm is based on an iterative cycle which computes in each cycle one term of the series:

$$\log x = \left(\frac{x-1}{x}\right) + \frac{1}{2}\left(\frac{x-1}{x}\right)^2 + \frac{1}{3}\left(\frac{x-1}{x}\right)^3 + \ldots\ldots \quad x \geq \frac{1}{2}$$

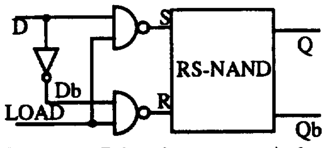
The requirement of x≥1/2 is not a limitation, since we evaluate the logarithm for arguments larger than 1.

4: Applications

Two applications have been carried out to check the effectiveness of the implemented model. The first is the implementation of a D-latch containing an RS-NAND latch as the memory element. The second tries to demonstrate that a bistable in metastability can be the cause of error when its metastable state is read by two other bistables and each reads a different value.

4.1: Model of a D-type latch

We are going to construct a D-type latch model using the model of the RS-NAND latch presented previously. The objective is to check if the descriptive model of the metastability is valid when we include the RS latch in another circuit. For this we will use the structure of the



L	D	Q
0	-	q
1	D	D

Figure 8. D-Latch composed of an RS-Latch.

D-latch shown in Figure 8. The code implemented to describe the D latch is shown in Figure 9. The operation mode is simple: input data (D) passes to output (Q) when condition LOAD=1 is true and there is a variation of any signal present in the wait clause (LOAD, D, Db).

```
entity latchd is
 port(D,LOAD: in MVL; QQb: out MVL_vector (1 downto 0));
end latchd;
architecture logical of latchd is
component latchrsnand
  generic(tpn,tsetup,tau,Twfil,Twmin: time);
  port (R,S:in MVL;QQb:out MVL_vector (1 downto 0));
end component;
signal Db,R,S:MVL:='0';
for all: latchrsnand use entity WORK.latchrsnand(logi-
cal);
begin
C1:latchrsnand generic map (4 ns, 4 ns, 1.8 ns, 1 ns, 4
ns) port map (R,S,QQb);
process
begin
          wait on D,Db,LOAD;
          Db<=not D after 1 ns;
          R<=LOAD nand Db after 1 ns;
          S<=LOAD nand D after 1 ns;
          end process;
end logical;
```

Figure 9. Model for a D-Latch using the proposed RS-latch.

Since the input inverter makes R and S appear permanently complemented, the D latch may only enter metastability provoked by a runt pulse in the RS latch. Simulation results are included in Appendix A showing how D-latch enters in metastability and at the same time, how the behavior is close to that expected.

4.2: Error caused by metastability

A case presenting the metastability as a cause of propagating errors is the circuit of Figure 10. This shows how the output of an RS latch (L1) drive the two synchronous RS latches (L2 and L3). The presence of a small skew between the two load signals of the latches, together with the entry into metastability of latch L1 may cause latch L2 and L3 to capture different logic values. This is a classic example of the undesired effects of metastability.

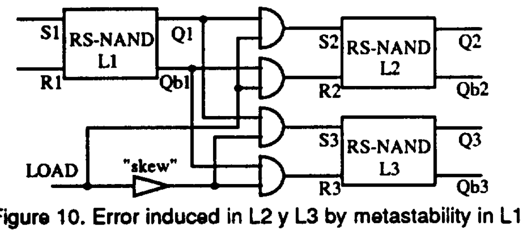


Figure 10. Error induced in L2 y L3 by metastability in L1.

5: Conclusions

A complete VHDL model of bistables has been presented. This model includes normal behavior and the so-called exponential model of its mestastable behavior. Its application to the analysis of circuits with timing problems allows not only to detect the problem itself (metastability) but also to know its evolution depending on the time restriction violations. Thus, the utility of the proposed model is its ability to perform realistic simulation of systems. To develop the model, we have faced some problems related to the treatment of TIME, but have been resolved via specific solutions. The core of the implementation is the model of the basic memory element (RS-NAND latch). Thus, the modelling of more complex bistables is quite straightforward, presenting a D-latch as an example. In a similar way, we have used the proposed model to show how a circuit with three latches can be a source of fatal errors caused by metastability transmission in one of its latches.

6: References

[1] Lipsett, R., Schaefer, C. and Ussery, C.: "VHDL: Hardware Description and Design", Kluwer Acad. Pubs. 1990.

[2] Berge, J.M., Fonkoua, A., Maginot, S. and Rouillard, J.: "VHDL Designers Reference", Kluwer Acad. Pubs. 1992.

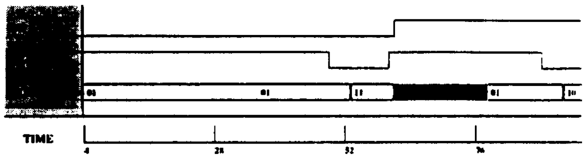
[3] Rosemberg, F. and Chaney, T.J.: "Flip-Flop Resolving Time Test Circuit", IEEE Journal of Solid-State Circuits, Vol. SC-17, No. 4, pp. 731-738. August 1982.

[4] Bellido, et. al.: "A New Faster Method for Calculating the Resolution Coefficient of CMOS Latches: Design of an Optimun Latch", Proc. of 26th ISCAS,pp.2019-2022. 1993.

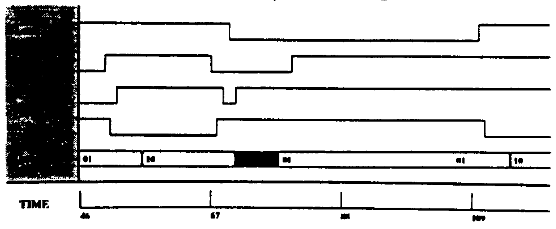
[5] Morin, L. and Li, H.F.: "Design of Synchronisers: a Review", IEE Proceedings-E, Vol. 136, No.6, pp. 557-564. Nov. 1989.

APPENDIX A

Simulation results. Parameters of RS-latch are:
tpn=4ns,tsetup=4ns,tau=10ns,Twfil=1ns,Twmin=4ns



RS-NAND latch under setup violation. g = 1 ns, tr = 17 ns.



D-latch (R-Runt pulse violation). tr = 7 ns.