

# Note on using computer simulations in teaching introductory digital electronics courses

C. G. Stafford Baines

*Department of Physics, University of Durban–Westville, Durban 4001, South Africa*

Michael E. Bacon<sup>a)</sup>

*Department of Physics, Thiel College, Greenville, Pennsylvania 16125*

*(Received 13 May 1996; accepted 10 October 1996)*

---

In teaching modern digital electronics to physics students it is becoming increasingly important that both student and instructor learn to use simulation programs. We have found that many of the exercises that recent textbooks recommend for simulation including those of an elementary nature do not work. Using the SR flip-flop as an example, we indicate some of the pitfalls encountered. In an attempt to shorten the learning curve, we highlight the source of the problem and show how the problem can be solved. © 1997 American Institute of Physics. [S0894-1866(97)01701-X]

---

## INTRODUCTION

The modern trend in teaching electronics is to complement lectures on theory and laboratory exercises with the computer simulation of circuits. This trend is reflected in the growing number of textbooks<sup>1–4</sup> that encourage the use of circuit simulations.

At the University of Durban–Westville and at Thiel College we have adopted the PSPICE<sup>5</sup> based evaluation package<sup>6</sup> from Microsim Corporation for use in the required electronics courses for applied physics and physics majors. We believe that it is important that our students develop a familiarity with the widely used PSPICE protocol and be familiar with a simulation package that has a commercial counterpart. In addition, the authors have found the complementing of theory and laboratory work by computer modeling to be a powerful pedagogic tool and our students are encouraged to model as many circuits as possible during laboratory sessions and also on their own time.

In this article we will describe our experiences in using the Microsim evaluation package to simulate the operation of the SR flip-flop and focus on the quirks (bugs) we encountered. At this point it is perhaps worth quoting the view expressed in Ref. 3 that “...the increasing use of CAD means that, instead of spending time fighting with soldering irons and test clips, many designers can now spend their time fighting with buggy CAD programs...” We hope, however, that this article will save others significant time and effort.

The simulation packages the authors have used generally work well. So well that instructor and student alike may have a tendency to lose sight of the fact that these computer based aids are “simulations” and not “real circuits.” One is therefore surprised when anomalies arise in relatively simple but pedagogically interesting circuits. Such is the case with the SR flipflop constructed from two

NOR gates that is the subject of this article.

Generally a discussion of this simple configuration of NOR gates is the starting point for describing flip-flop operation in digital electronics. Expanding on this basic structure leads ultimately to the edge triggered D and JK type flip-flops and a discussion of latches, counters, and registers.

Although a number of texts recommends that students simulate circuits, most do not indicate possible pitfalls in using the computer simulations. The one text<sup>2</sup> that we did find that mentioned the same problem with which we were confronted, gives no insight into the cause of the problem nor guidance on how to remedy it. Although we used the Microsim evaluation package for this study, the same anomalous behavior is exhibited by Logicworks<sup>7</sup> and Electronic Workbench.<sup>8</sup>

## I. THE SIMULATION

In introducing the SR flip-flop made from two NOR gates, Fig. 1 (top), one generally considers the possible outputs for Q and Qbar, given the four possible S,R inputs, and develops the truth table shown in Fig. 1 (bottom). The first input state is sometimes referred to as the memory state, since a transition to this state from either of the next two states listed in the truth table will hold or store the 0 (low) or 1 (high) that is present in these states (0,1 or 1,0). The last state (1,1) is often called the disallowed state, although it is a perfectly well-defined state. Perhaps a better name for this state is the undesirable state. It is, however, different from the others in that both Q and Qbar are 0 for this input state whereas for all other input states Q and Qbar are complements. In addition, and most important, it is a state to be avoided since, if this state is followed by the memory state, both inputs must change simultaneously (1,1 to 0,0). Since the inputs are unlikely to change simultaneously, the system will be in one or the other of the two states 0,1 or 1,0 for a brief time before it reaches the final memory state. For two arbitrary NOR gates it is impossible to predict

---

<sup>a)</sup>E-mail: mbacon@thiel.edu

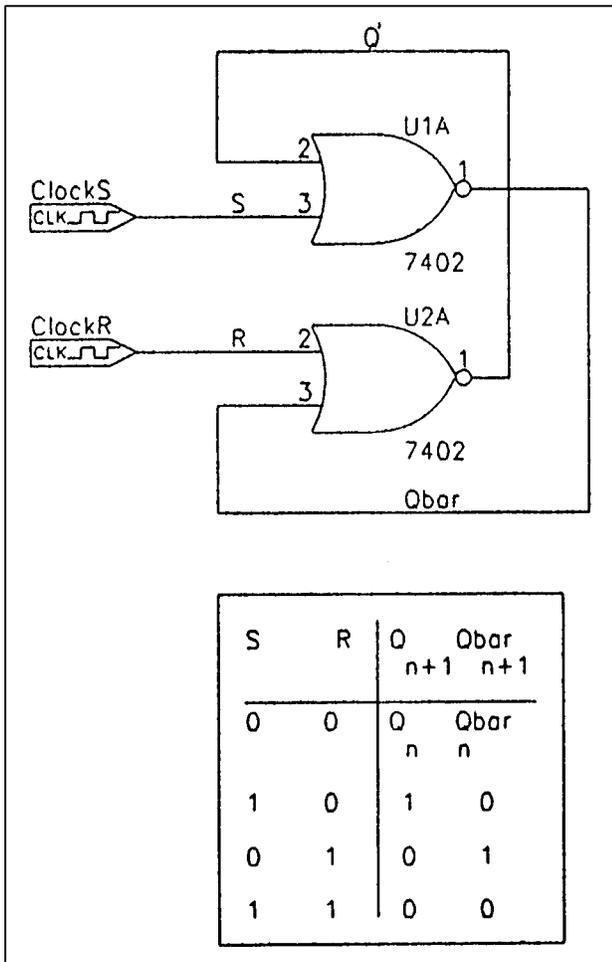


Figure 1. Top: SR flip-flop using NOR gates (Design Center: SCHEMATICS). Bottom: SR Truth Table.

ahead of time which of the two possibilities will occur. For this reason the 1,1 state should be avoided in practice.

In a laboratory exercise students typically investigate the behavior of the flip-flop using debounced logic switches for the inputs and a light emitting diode (LED) wired to the output to confirm the truth table. Simulating this using Log-icworks is very easy and the simulation reproduces the appropriate truth table in accord with experiment.

In attempting to simulate this simple circuit using the Microsim evaluation package, however, it is inconvenient to connect highs and lows to the inputs and monitor the output. It is much easier to clock the inputs with two clocks of differing periods so that all four states are generated. The circuit simulation can be set up using SCHEMATICS and analyzed by setting up a "transient" or time varying analysis, generating a netlist using PSPICE and finally using PROBE to display the input and output wave forms. When this is done, we obtain the data shown in Fig. 2 (top). In this simulation the undesirable transition from 1,1 to 0,0 occurs and causes what turns out to be unrealistic high frequency oscillatory behavior as seen in Fig. 2 (bottom). This strange behavior was noted by Savant<sup>2</sup> using a simulation program called MICRO-CAP II. As mentioned earlier, this unrealistic behavior also occurs if we use Logicworks or Electronics Workbench to simulate the same scenario.

Since experimentally it is somewhat inconvenient to use two clocks for the S and R inputs, and recognizing that if you clock both inputs with the same clock the same undesirable transition from 1,1 to 0,0 will occur, we then simulated the circuit shown in Fig. 3 (top) and obtained the results shown in Fig. 3 (bottom). This confirmed the simulations' oscillatory behavior when there is a transition from 1,1 to 0,0.

If the circuit shown in Fig. 3 (top) is assembled on a breadboard using two of the four NOR gates from the 74LS02 and a 100 kHz clock, one of the outputs oscillates

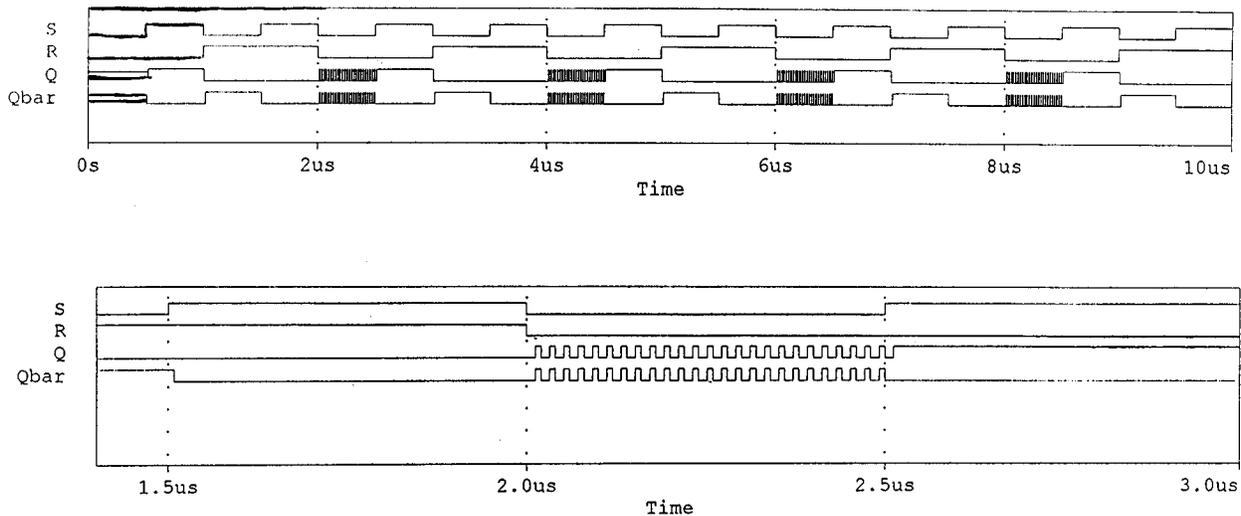


Figure 2. Top: Simulation results for the NOR gate SR flip-flop using Design Center (PSPICE, PROBE). Bottom: Expanded view of the shaded region of top panel showing oscillations in the simulation.

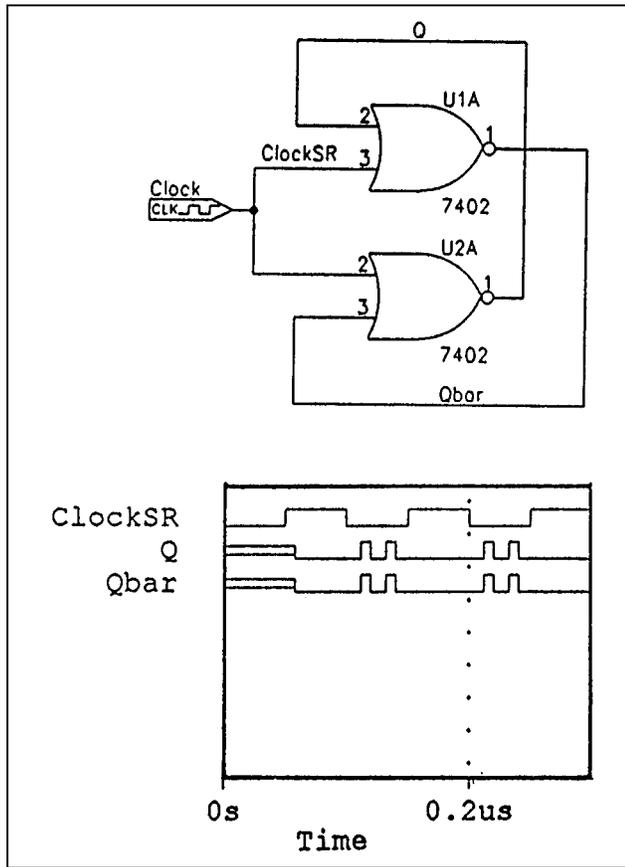


Figure 3. Top: SR flip-flop driven by a single clock using Design Center (SCHEMATICS). Bottom: Simulation results for a SR flip-flop driven by a single clock (Design Center: PSPICE, PROBE).

at the clock frequency while the other output stays level at 0. The high frequency oscillation exhibited by the simulation does not occur in practice.

The behavior illustrated by the real circuit is understandable on the basis of the asymmetry in the propagation delay for the two NOR gates. When the clock makes the transition from 1,1 to 0,0, one NOR gate has a slightly slower propagation delay relative to the other and in our case this results in a transition to the memory state through the state 0,1, causing Q to oscillate at the clock frequency and Qbar to remain at zero.

Herein, of course, lies the key to the failure of the simulation to predict the real world behavior. In the simulations the default value for the propagation delay of each NOR gate is exactly the same. Fortunately many state-of-the-art simulation programs allow for modification of certain model parameters and in the Microsim evaluation package the propagation delay may be set to min, max, or typical. If we introduce this asymmetry into our simulation and thereby insure nonequality of the propagation delay in the two NOR gates, we obtain the simulated results shown in Fig. 4. This is in accord with the experimental observations mentioned above. A simple way of introducing a difference in the propagation delay in the two input signals if the propagation delay of the gates cannot be adjusted is to put a buffer in one arm.

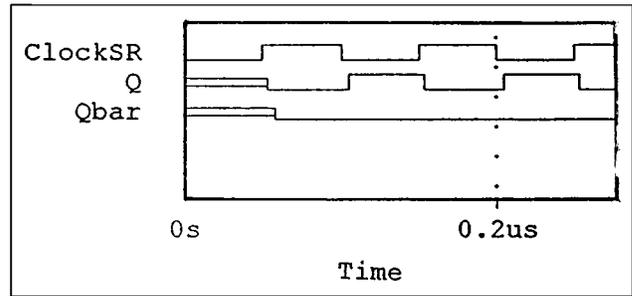


Figure 4. Simulation results with one NOR gate having a typical propagation delay and the other having a minimum propagation delay (Design Center: PSPICE, PROBE).

In a similar way the other two simulation programs that we have tried may be adjusted to give realistic results. We note, however, that in using Logicworks the propagation delay times of individual NOR gates taken from the GATE LIBRARY can be modified separately but that, if instead the Quad NOR IC, 7402 is used, the propagation delay for the gates on the IC cannot be adjusted individually and the simulation fails to produce the experimentally observed results, giving instead the unrealistic oscillatory behavior discussed above.

The oscillatory behavior obtained in the simulation using two identical NOR gates can be explained in terms of the equality of the (output) 0 to 1 propagation delay  $t_{LH}$  of the two gates, and also the equality of the 1 to 0 propagation delay  $t_{HL}$ .

When ClockSR [Fig. 3 (bottom)] makes the transition from 1 to 0, yielding inputs to the two NOR gates of 0,0, a time  $t_{LH}$  elapses before the outputs of the two NOR gates make the transition from 0 to 1. The inputs to the two NOR gates are both then 1,0. After a time  $t_{HL}$  the outputs make the 1 to 0 transition resulting in the inputs to the two NOR gates again being 0,0. This pattern repeats as long as ClockSR is in the zero state. The result is an oscillation with period  $t = t_{LH} + t_{HL}$  and a duty cycle equal to  $t_{HL}/t$ .

In the simulation the default values (typical) for  $t_{LH}$  and  $t_{HL}$  are 12 and 8 ns, respectively. Measurements on the simulation outputs [Fig. 3 (bottom)] confirmed that the period is 20 ns and the duty cycle is 0.4 in agreement with the above equations.

## II. CONCLUDING REMARKS

The problem we have discussed in regard to simulating the SR flip-flop occurs for the cross coupled NAND gate configuration also. In addition it affects the simulation results that are obtained as one moves on toward developing more advanced flip-flop structures. For example, the clocked RS and some master-slave configurations show the same non-physical oscillatory behavior unless the propagation delays for the individual gates are adjusted so that they are not equal. On the other hand, the problem does not occur when one modifies the RS structure to make a level-triggered D type or D type latch. In this case, of course, the disallowed state never occurs and the problem that we have addressed in this article is not relevant.

## ACKNOWLEDGMENT

The authors would like to express their thanks to the referees and in particular to the referee who suggested the inclusion of the explanation of the oscillations and the length of the duty cycle described in the three paragraphs preceding the concluding remarks.

## REFERENCES

1. D. J. Crecraft, D. A. Gorham, and J. J. Sparks, *Electronics* (Chapman and Hall, New York, 1993).
2. C. J. Savant, Jr., M. S. Roden, and G. L. Carpenter, *Electronic Design Circuits and Systems*, 2nd ed. (Benjamin/Cummings, Redwood City, CA, 1991), p. 739.
3. J. F. Wakerly, *Digital Design Principles and Practices*, 2nd ed. (Prentice-Hall, Englewood Cliffs, NJ, 1994), p. 764.
4. G. C. Loveday, *Digital and Analogue Electronics for HNC* (Longman Scientific and Technical, Harlow, Essex, UK, 1993).
5. SPICE (simulated program with integrated circuit emphasis). This program was developed at the University of California, Berkeley, and was first released in about 1973. One derivative is PSPICE from Microsim Corporation.
6. The Design Center, Microsim Corporation, 20 Fairbanks, Irvine, CA 92718.
7. Logicworks, Capilano Computing Systems, Ltd., 406-960 Quayside Drive, New Westminster, BC V3M6G2, Canada.
8. Electronics Workbench, Interactive Image Technologies, 908 Niagara Falls Blvd. No. 068, North Tonawanda, NY 14120-2060.