Modeling and Verification of D-flip flop using PROMELA and SPIN

NEPHAS MUFUTUMARI§
Zimbabwe Open University
Zimbabwe

TAURAYI RUPERE
University of Zimbabwe
Zimbabwe

ABSTRACT

The formal specification language PROMELA and the SPIN model verification tool are used for formal specification and verification of correctness requirements for systems of concurrently executing processes. Extensive literature exists on the application of Spin and PROMELA in software systems, especially communication protocols modeling and verification. However, little literature exists on the use of PROMELA and SPIN in hardware systems specification and verification. The research looked at the use PROMELA to specify the delay flip flop (D-flip flop) and the use of SPIN to verify the implemented design. The results showed that the SPIN model verifier can pick up some timing constraint, dynamic and static hazards in sequential circuits. A variety of captured errors are considered to be not harmful in a real practical application. However, time constraint, which is not a notion in SPIN, need to be dealt with in embedded systems design and safety critical systems to deal with the sequential systems.

Categories and Subject Descriptors: B.6 (Logic Design]: Design Styles and Design Aids- Sequential Circuits-Simulation and Verification

Key Words: PROMELA, SPIN, SR-Flip-Flop, D-Flip-Flop, State Diagram, Hazard, Modeling, Verification

IJCIR Reference Format:

Nephas Mufutumari and Taurayi Rupere. Modeling and Verification of D-flip flop using PROMELA and SPIN. International Journal of Computing and ICT Research, Vol. 7 Issue 2, pp 48-53. http://www.ijcir.org/volume7-issue 2/article 4.pdf

§ Author's Address: Nephas Mufutumari - Zimbabwe Open University, 7th Floor Stanley House, Corner First Street/J. Moyo Avenue - Harare, Zimbabwe
Taurayi Rupere - University of Zimbabwe, Computer Science Department, P.O. Box MP 167 - Harare, Zimbabwe

International Journal of Computing and ICT Research, ISSN 1818-1139 (Print), ISSN 1996-1065 (Online), Vol. 7, Issue 2, pp. 48-53, December 2013.

1.0 INTRODUCTION

Major designs of electronic systems tend to be modular. In some situations the modules are designed and constructed by different teams or companies. Although the modules themselves maybe thoroughly independently tested and verified, problems may arise when they are interconnected. It is commonly accepted that design for testability (DfT) could eliminate most of the systems design and functionality problems. The goals for DfT are:

- "Minimizing costs of system production
- Minimizing system test complexity : test generation and application
- Improving quality
- Avoiding problems of timing discordance or block nature incompatibility." (Mylnek D. Web Ref, 2004)

One well established way of achieving these objectives is to use formal systems and hardware design languages (HDL) to specify and verify the designs (Glunz and Venzl 1992; Shimizu et al. 2000; Maissel. and Ofek. Web Ref. 2004).

PROMELA and SPIN were originally meant for design and verification of computer communication protocols (Holzmann, 1991). The use of these tools for protocol verification has been extensively reported (Brinksma and Mader, 2000; Jensen et al. 1996 ; Martel and Gengler, 2000; Merino and Troya, 1996 ; Najm and Olsen, 1996; Shanbhag and Gopinath 2001). Applications of PROMELA/SPIN especially in formal software systems modeling and verification have also been reported (Ball and Rajamani 2001; M and Miller 2001; Nakajima and Tamai 2001; Stoller and Liu 2001). An application of PROMELA and SPIN as tools for the specification and verification of combinational circuits is reported in (Rahardjo, 1995) while (Van Eijk, 1997) reports their usage in the verification of relay circuits. In this paper, the researchers investigated how systems engineering process optimization transcends engineering disciplines by applying PROMELA and SPIN to modeling and verification of a sequential circuit.

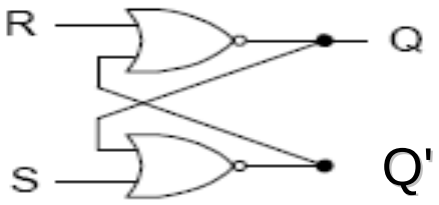
2. 0 BACKGROUND

2.1 Theoretical Behaviour of flip flop

The output of a sequential circuit is characterized by the current input and the current output state. Every sequential circuit uses feedback, stable states and a way to establish the required states so that it can be stable. The fundamental sequential circuit is the set-reset (SR) flip-flop.

2.2 SR Flip-Flop

The digital logic diagram for an SR flip-flop, implemented using NOR gates, is shown in figure 1. The circuit has inputs S and R with an outputs Q and Q', where Q' is always meant to be the inverse of Q.



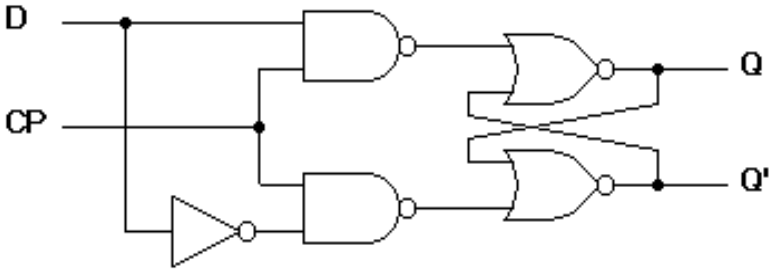
Q	S	R	Q^+
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Undefined
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Undefined

Figure 2: SR Flip Flop Characteristic table

Figure 2 shows the characteristic table for the circuit. For the input S=R=1, the behavior of the circuit is not deterministic and therefore is a hazardous state. The possible states are represented by two nodes. The inputs S and R are represented by the edges; labels on the edges are the values of S and R in that order. Only valid inputs are shown, therefore the combination S=R=1 does not appear because it is not an input that causes a valid state transition.

2.3 D flip-flop

A way of dealing with the SR flip-flop race condition hazard is to allow just one single input. This is implemented in a common circuitry configuration called the D-flip-flop. The diagram and state diagram for the D-flip-flop are shown in figures 3 and 4 below. Using an inverter between the two non clock inputs to the two NAND gates guarantees that the hazardous S=R=1 condition is not expected to occur.



Q D	Q(t+1)
0 0	0
0 1	1
1 0	0
1 1	1

Figure 3. D Flip-flop **Figure 4. D Flip-flop Characteristic Table**

3.0 MATERIALS AND METHODS

The PROMELA model of the fundamental mode D flip flop was presented based on the non-deterministic property of SPIN model instantiation. The Set and Reset states were modeled as processes. Since this is a fundamental mode circuit, the input change was also modeled as a process that may change either of the inputs in a non-deterministic manner.The model was verified with the XSpin basic options verifier with the claim option set. Other verifications with the search depth altered were performed. The PROMELA model was simulated at random using XSpin with a seed value of 1.This was then compared with the theoretically expected Q output and the model behavior of Q output for the D flip flop.

3.1 PROMELA model of D flip-flop simulation

```
/*a simulation model of the D-flip flop*/
bit out2, out1, new_out1, new_out2, d_input=0, clock=0;
#define nor(input1,input2,output) (output ==!(input1||input2))
active proctype set()
{ do      ::
atomic   {nor((d_input&&clock),out1,out2);          nor((!d_input&&clock),out2, out1);          new_out2=0;
new_out1=1;                          if
::(nor(!(!d_input&&clock),out2, out1))&&(!(nor(!(d_input&&clock),out1,out2)))
                    fi;
                    }od  }

active proctype reset()
{do ::
atomic { new_out2=1; new_out1=0; nor(!(d_input&&clock),out1,out2); nor(!(!d_input&&clock),out2, out1);
                          if
          ::(!(nor(!(!d_input&&clock),out2,out1)))&&(nor(!(d_input&&clock),out1,out2))
                    fi;
                    }
od
```

```
                    }
active proctype input_change()
        {do
                    ::timeout->
atomic { out1=new_out1; out2=new_out2;
                    if
        ::d_input=1-d_input
::clock = 1-clock
                    fi;
        }
        od
        }
never {  do
         ::((d_input==1)&&(clock==1)&&(new_out1==0)) ->break
        ::((new_out1==1)&&(new_out2==1))->break
        ::((d_input==0)&&((clock==0)||(clock==1))&&(new_out1==1))->break
        ::(((d_input==1)||(d_input==0))&&(clock==0)&&(new_out1==1))->break
        ::else -> skip
        od }
```

4.0 RESULTS

4.1 Verification Results

The following output was generated. Other verifications with the search depth altered were performed and they showed that all the claim conditions were violated.

```
pan: claim violated! (at depth 8)
pan: wrote pan_in1.trail
(Spin Version 4.2.0 -- 27 June 2004)
Warning: Search not completed

Full statespace search for:
        never claim         +
        assertion violations        - (disabled by -A flag)
        cycle checks        - (disabled by -DSAFETY)
        invalid end states - (disabled by never claim)

State-vector 28 byte, depth reached 8, errors: 1
    3 states, stored
    0 states, matched
    3 transitions (= stored+matched)
    5 atomic steps
hash conflicts: 0 (resolved)

Stats on memory usage (in Megabytes):
0.0     equivalent memory usage for states (stored*(State-vector + overhead))
```

4.2 Random Simulation Results

The model was simulated at random using XSpin with a seed value of 1. The figures below show comparison of the theoretically expected Q output and the model behavior of Q output for the D flip flop.

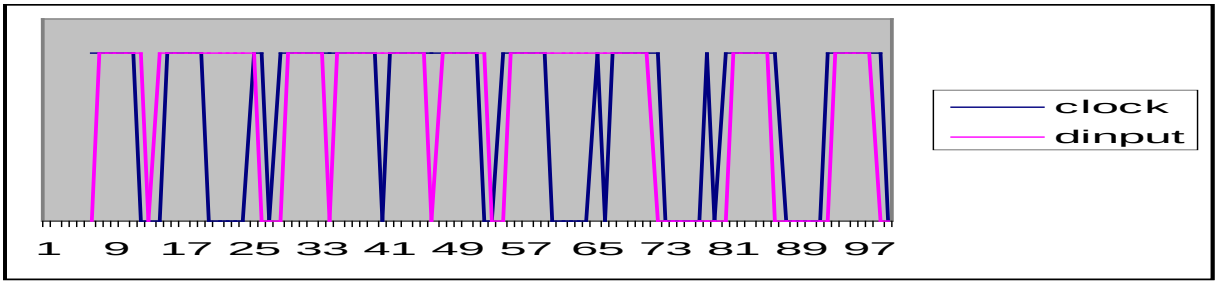


Figure 5. Simulation control values for clock and d input

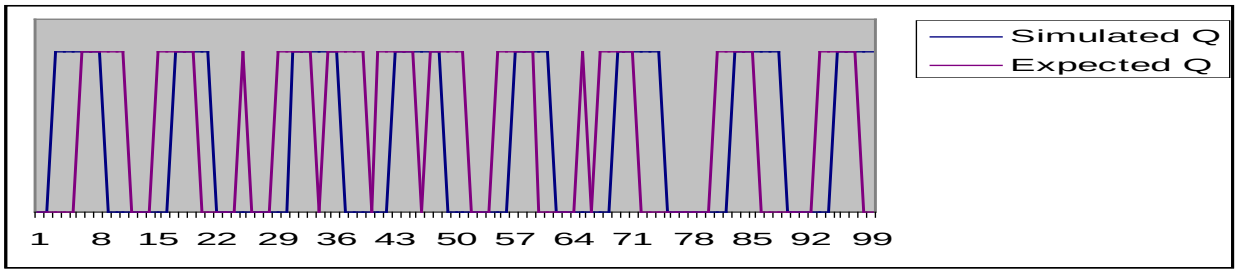


Figure 6. Comparison of the simulated and theoretical behaviors of the D-Flip Flop

5.0 DISCUSSION AND FINDINGS

The PROMELA model has a *never claim* which is a direct consequence of the characteristic behavior of the D-flip flop. The SPIN verification shows that the *never claim* is violated from as low a depth as 8. The violations are also demonstrated by the simulation results which show several points where the simulated output does not agree with the stated characteristics of the D-flip flop as shown in figure 6. This shows that the statements in the *never claim* fail to hold at least once. Thus there will be at least one situation when the condition: D=1& CP=1 =>Q(t+1) =1 fail to hold. This is likewise for all the other entries in the characteristic table. These variations are expected in electronic circuit design. They are classified as timing constraint, static and dynamic hazards.

The timing constraints simply mean that it takes finite time for a gate to change from one state to another. Thus should an input change during a current output transition it is very possible that it may not be registered. This problem is compounded when a logic circuit made up of several logic gates is driven faster than the response speed of at least one of the gates. This may lead to a situation where the path that contains the slower gate registers a change well after the faster path.This is one reason why all electronic components have a response speed specification from the manufacturer to solve this problem.

The static hazard occurs when the change of a single variable causes a momentary change in other variables in the circuit. This may be caused by voltage surges. On the other hand certain outputs, especially in flip flops may go through multiple transitions for a short time causing dynamic hazards. Thus instead of the output switching as: $\mathbf{0 \rightarrow 1}$ it may switch as $\mathbf{0 \rightarrow 1 \rightarrow 0 \rightarrow 1.}$ Practical electronic circuits generally have redundant terms to solve the static problems.

6.0 CONCLUSION

Sequential circuital components such as the D-flip flop can be analysed using SPIN in great detail and a variety of errors captured. So clearly the PROMELA specification language and the model verifier and simulator tool, SPIN, can be used to check the correctness of the models. This has the advantage that models having substantial differences can be easily compared against each other' correctness for behavioural similarities.

A major limitation of this approach to electronic circuit design is that the model verifier picks some errors that may be considered to be not harmful in real practical applications. More research is however needed to deal with the issue of time, which notion is not available in SPIN but is still being developed. Further applications in embedded systems design and safety critical systems are envisaged.

7.0 REFERENCES

BALL T. B. AND, RAJAMANI S. K. Automatically Validating Temporal Safety Properties of Interfaces. In Dwyer M.B (Ed.): Model Checking Software, 8th International SPIN Workshop, Toronto, Canada, May 19-20, 2001, Proceedings. Lecture Notes in Computer Science 2057 Springer 2001, ISBN 3-540-42124-6 163-182 (pp103-122)

BRINKSMA E. AND MADER. 2000. A Verification and Optimization of a PLC Control Schedule, Lecture Notes in Computer Science Volume 1885, Springer Verlag, ISSN 0302-9743 (pp. 73-92)

GLUNZ W. AND VENZL G. 1992. Hardware design using CASE tools. In Halaas A. and Denyer P.B., editors, VLSI 91.

HOLZMANN G.J. 2004. The Spin Model Checker, Primer and Reference Manual. Addison-Wesley

HOLZMANN G. J. 1991. Design and validation of computer Protocols. Prentice Hall, New Jersey. ISBN 0-13-539925-4

JENSEN H. E., LARSEN K. G., AND SKOU A. 1996. Modeling and analysis of a collision avoidance protocol using Spin and Uppaal. In Grégoire JC, Holzmann G.J and Peled D.A editors The Spin Verification System" volume two. DIMACS

MAISSEL L. I. AND OFEK H. WEB REF. 2004. http://www.research.ibm.com/journal/rd/285/ibmrd2805G.pdf

MARTEL M. AND GENGLER M. 2000. Communication Topology Analysis for Concurrent Programs. Lecture Notes in Computer Science Volume 1885, Springer Verlag, ISSN 0302-9743 (pp. 265-286)

M C. AND MILLER A. Using SPIN for Feature Interaction Analysis - A Case Study. In Dwyer M.B (Ed.): Model Checking Software, 8th International SPIN Workshop, Toronto, Canada, May 19-20, 2001, Proceedings. Lecture Notes in Computer Science 2057 Springer 2001, ISBN 3-540-42124-6 (p143-162)

MERINO P. AND TROYA J. 1996. Modeling and verification of the MCS layer with Spin. In Grégoire JC, Holzmann G.J and Peled D.A editors The Spin Verification System" volume two. DIMACS

MYLNEK D. WEB REF 2004. http://lsiwww.epfl.ch/LSI2001/teaching/webcourse/ch08/ch08.html

NAJM E. AND OLSEN F. 1996. Protocol verification with reactive Promela/Rspin. In Grégoire JC, Holzmann G.J and Peled D.A editors The Spin Verification System" volume two. DIMACS

NAKAJIMA S AND TAMAI T. Behavioural Analysis of the Enterprise JavaBeans™ Component Architecture. In Dwyer M.B (Ed.): Model Checking Software, 8th International SPIN Workshop, Toronto, Canada, May 19-20, 2001, Proceedings. Lecture Notes in Computer Science 2057 Springer 2001, ISBN 3-540-42124-6 (p163-182)

RAHARDJO B. SPIN as a hardware Design Tool. SPIN95, the First SPIN Workshop NRS-Télécommunications, Montréal, Quebec On 16 October 1995 (http://spinroot.com/spin/Workshops/ws95/papers.html)

SHANBHAG V.K., GOPINATH K. A SPIN-Based Model Checker for Telecommunication Protocols. In Dwyer M.B (Ed.): Model Checking Software, 8th International SPIN Workshop, Toronto, Canada, May 19-20, 2001, Proceedings. Lecture Notes in Computer Science 2057 Springer 2001, ISBN 3-540-42124-6 (p 252-271)

SHIMIZU K., DILL D. L., AND. HU A. J. 2000. Monitor-Based Formal Specification of PCI. In Hunt W. A. and Johnson S. D., editors, *Formal Methods in Computer-Aided Design*, volume 1954 of *LNCS*, pages 335–352,.

STOLLER S. D AND LIU Y. A. Transformations for Model Checking Distributed Java Programs. In Dwyer M.B (Ed.): Model Checking Software, 8th International SPIN Workshop, Toronto, Canada, May 19-20, 2001, Proceedings. Lecture Notes in Computer Science 2057 Springer 2001, ISBN 3-540-42124-6 (pp192-199)

VAN EIJK P. Verifying Relay Circuits using State Machines. Proceedings of SPIN97, the Third SPIN Workshop. Twente University, Enschede, The Netherlands 5 April 1997 (http://spinroot.com/spin/Workshops/ws97/papers.html)