Report Number: WUCSE-2005-52

2005-11-09

Design of an Interlock Module for Use in a Globally Asynchronous, Locally Synchronous Design Methodology

U. G. Swamy, J. R. Cox, G. L. Engel, and D. M. Zar

As the number of transistors on a single integrated circuit approach a billion, the problems of clock distribution, power consumption, multiple clock domains, meeting timing requirements, and reuse of subsystem designs grow ever more difficult. Coordinating a billion transistors with the present design methodologies will require hundreds of years of engineering time. A new design methodology is needed. The GALS (Globally Asynchronous Locally Synchronous) approach, that blends clockless and clocked subsystems is a strong contender.

Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

2005-52

Design of an Interlock Module for Use in a Globally Asynchronous, Locally Synchronous Design Methodology

Authors: U. G. Swamy, J. R. Cox, G. L. Engel, D. M. Zar

November 9, 2005

Abstract: As the number of transistors on a single integrated circuit approach a billion, the problems of clock distribution, power consumption, multiple clock domains, meeting timing requirements, and reuse of subsystem designs grow ever more difficult. Coordinating a billion transistors with the present design methodologies will require hundreds of years of engineering time. A new design methodology is needed. The GALS (Globally Asynchronous Locally Synchronous) approach, that blends clockless and clocked subsystems is a strong contender.

Design of an Interlock Module for Use in a Globally Asynchronous, Locally Synchronous Design Methodology

U. G. Swamy[1], J. R. Cox[2], G. L. Engel[1], D. M. Zar[2]

[1]*Department of Electrical and Computer Engineering, VLSI Design Research Laboratory, Southern Illinois University Edwardsville, Illinois, USA, 62025*

[2]*Department of Computer Science and Engineering, Washington University in Saint Louis, Illinois, USA, 63130*

Introduction

As the number of transistors on a single integrated circuit approach a billion, the problems of clock distribution, power consumption, multiple clock domains, meeting timing requirements, and reuse of subsystem designs grow ever more difficult. Coordinating a billion transistors with the present design methodologies will require hundreds of years of engineering time. A new design methodology is needed. The GALS (Globally Asynchronous Locally Synchronous) approach [1], [2] that blends clockless and clocked subsystems is a strong contender.

Blended Methodology

The proposed blended methodology utilizes multiple, independently clocked domains, restartable crystal clocks [3], [4] and asynchronous control elements to sequence interactions between processors in the various clocked domains. Combining a clock generator standard cell with standard cells for macromodular control elements [5] will provide the designer with the opportunity to blend simple asynchronous control networks with clocked subsystems without the need to meet global timing constraints. Additional advantages are freedom from the possibility of synchronizer failure [6], superior local timing accuracy, and mathematical tools for verifying correct sequence behavior. A key macromodular element in this methodology is the interlock [5], an element that ensures mutual exclusion between concurrent requests for a shared resource.

Interlock Design

The Petri net [7] model of the interlock and its environment is shown in Figure 1. The place e_t belongs to one remote processor and e_b to another. Requests from these processors are controlled by the interlock which provides access to places h_t and h_b belonging to the shared resource.

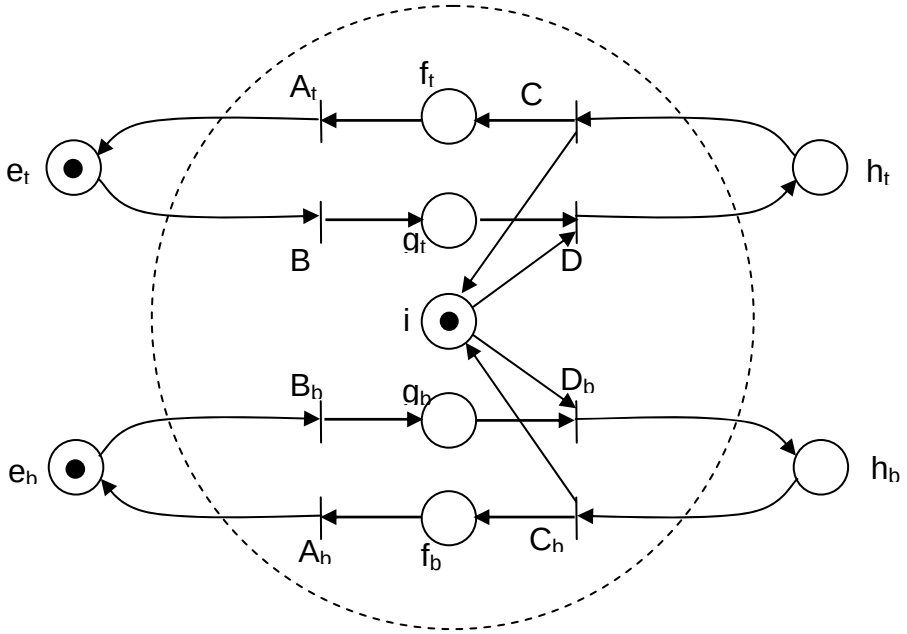

Figure 1: Petri net model of interlock and its environment

All Petri net transitions (A_t, B_t, C_t, D_t, A_b, B_b, C_b and D_b) in Figure 2 represent logic level transitions [8]. The firing of transition B_t moves the token from e_t to g_t. This enables transition D_t, because its two input places contains tokens, and D_t in turn fires. The firing of D_t grants the shared resource to the "t" processor. Thus, the two input places of D_t are emptied, and a token is placed in the shared resource place, h_t. Once use of the resource is completed, a token is returned to node 'i' by firing transition C_t. A completion signal is sent to the requesting processor only after the release of the shared resource.

If the two processors should simultaneously request the shared resource, both D_t and D_b are enabled and the interlock must go into an arbitration state (an unstable state) where it *resolves* the conflict between the two contenders. Only one of the outputs will transition, thereby allocating the resource to one of the processors.

The design methodology for the circuit to implement the interlock is detailed in [8]. The resulting logic equations are

$$db' = bb \cdot \overline{(ct \oplus dt)} + cb \cdot (ct \oplus dt)$$
$$dt' = bt \cdot \overline{(cb \oplus db)} + ct \cdot (cb \oplus db)$$

These equations describe the two output variables db' and dt' inferred from the Petri net reachability graph. The interlock circuit is shown in Figure 2. When inputs to the interlock transition simultaneously, the interlock exhibits oscillatory behavior while trying to resolve the conflict. It is necessary to detect when this metastable condition is present.

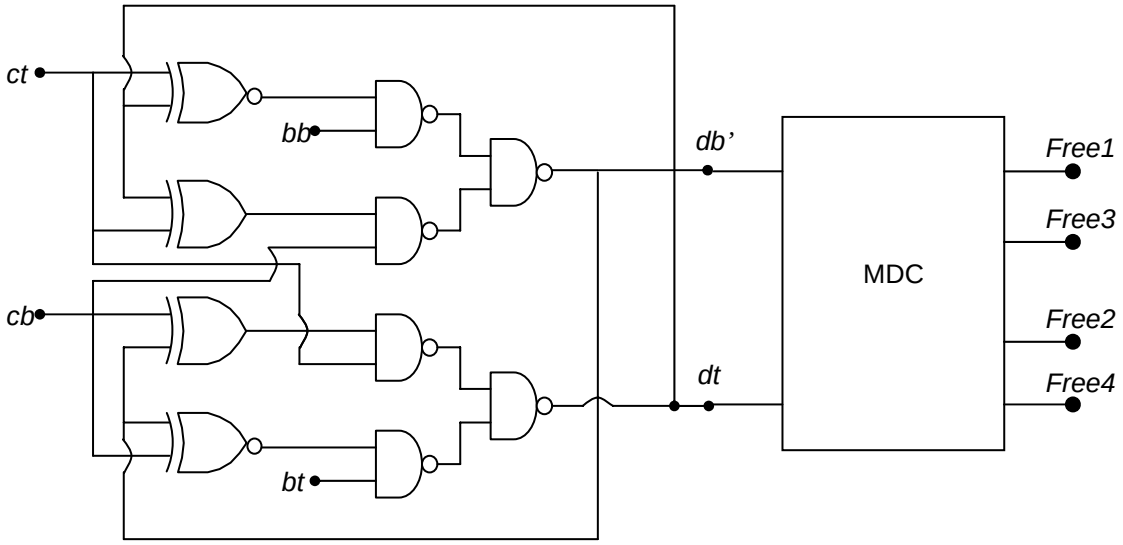

Figure 2: Schematic of interlock module with metastability detection circuit (MDC)

The metastability detection circuit (MDC) shown in Figure 3 was derived from a circuit described in [9]. The outputs (*Free1*, *Free2*) are held low whenever *db'* and *dt'* are equal; the outputs (*Free3, Free4*) are held low when *db'* and *dt'* are not equal. NFETS M_1 and M_6 are weak pulldowns and were added for the cases where the output nodes are discharged through either PFET M_2 or M_4. The transmission gate equalizes the inverter delay.

Metastability detection in the interlock is complicated by the fact that metastability could be present when the outputs db' and dt' are similar or when the outputs db' and dt' are different. These two cases can be distinguished independent of the possibly anomalous behavior of the outputs by examining the interlock completion signals *ab* and *at* (delayed versions of *cb* and *ct*). That is, if $ab = at$, metastability is present if $db' \cong dt'$ and if $ab \neq at$, metastability is present if $db' \neq dt'$.

The completed interlock circuit is presented in Figure 4. The rising edge of the stable signal, *S*, clocks FF_1 and FF_2 to copy both *dt'* and *db'* to their delayed and metastability-free versions *dt"* and *db"*. This circuit produces delayed and stable versions of the outputs *dt'* and *db'* upon every transition for either of these outputs.

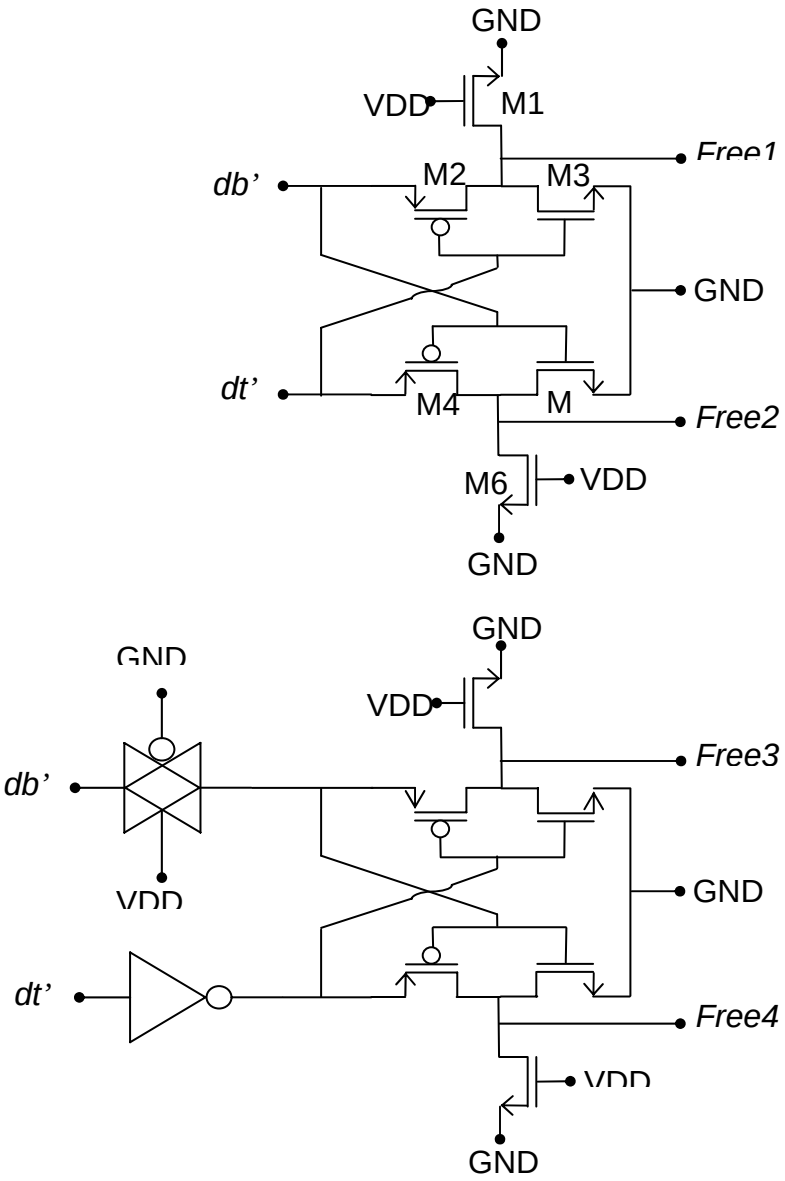

Figure 3: Metastability Detection Circuit (MDC)

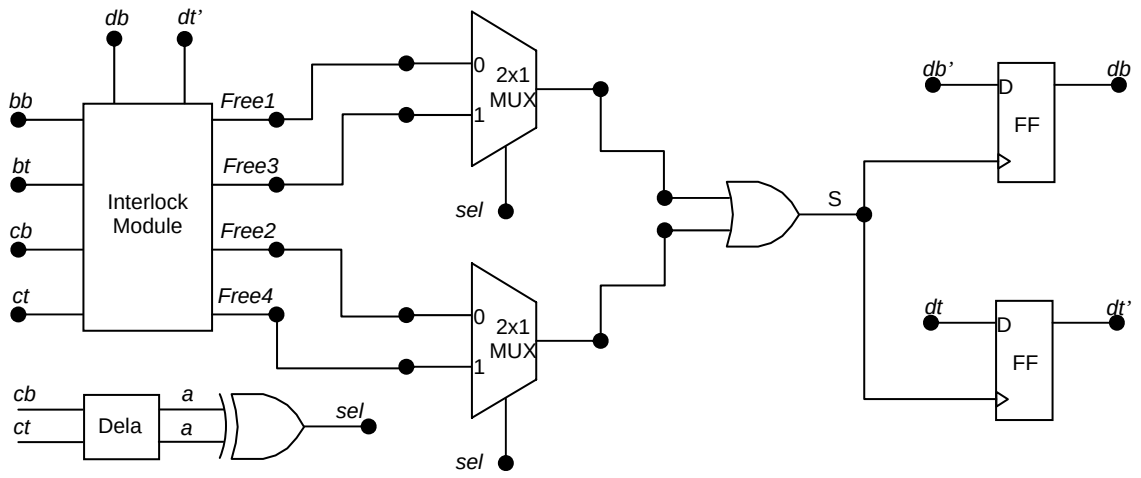

Figure 4: Circuit that produces stable version of interlock outputs

Simulation Results

Verification of the performance of the interlock was done using Spectre. The target technology is the TSMC n-well 0.25 μm process. There are four critical cases in the reachability graph where the inputs transition simultaneously. Simulation results presented in Figure 5 demonstrate how the interlock resolves the conflict between the two contenders successfully in one of those cases. The inputs *cb* and *ct* are not shown because they are in the low state throughout the simulation.

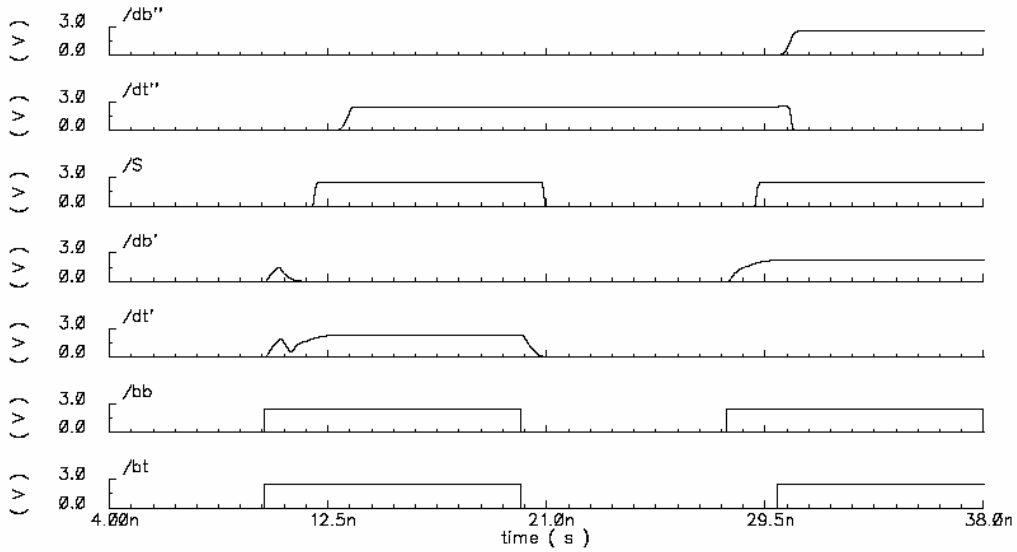

Figure 5: Interlock when in state '00' and the input requests bb and bt change simultaneously

Prior to 10 ns, the inputs (*bb*, *bt*, *cb*, and *ct*) and the outputs (*db'* and *dt'*) are stable. At 10 ns both *bb* and *bt* change. The interlock is forced to decide which requesting module will receive the shared resource. Both outputs *db'* and *dt'* pulse high then low.

When, both *db'* and *dt'* are high, the interlock is in an arbitration state as mentioned earlier; when both are low, then it is in "don't care" state as predicted in [8]. Output *db'* resolves high and *dt'* resolves low; outputs *db"* and *dt"* are free of oscillations. Also, at 28 ns, *bb* clearly transitions before *bt* and as expected the "b" processor is granted use of the shared resource. Similar results are obtained for the other three arbitration states.

Acknowledgement

The authors would like to thank Trong Wu, Department of Computer Science, Southern Illinois University Edwardsville, for providing valuable information regarding Petri nets and to Sasi Tallapragada for his assistance in simulating the interlock.

References

[1] D.M. Chapiro, *Globally-Asynchronous Locally-Synchronous Systems*, Ph.D. thesis, Stanford University, Oct. 1984.

[2] K.Y. Yun and A. E. Dooply, "Plausible clocking based heterogenous systems", IEEE Transactions on VLSI Systems, vol. 7, no. 4, pp 482-487, Dec. 1999.

[3] J.R. Cox, "Can a crystal clock be started and stopped?" Appl. Math. Lett. Vol 1, No. 1, pp. 37-40, 1988.

[4] J. R. Cox and D. M. Zar, "An asynchronous crystal clock for use in GALS system," Poster presented at Clockless Computing Symposium, Washington University, 26 March 2004.

[5] S.M. Ornstein, M. J. Stucki and W.A. Clark, "A functional description of macromodules," Spring Joint Computer Conf., AFIPS Proceedings, Vol. 30, Thompson Books, Washington D. C., pp. 357-364, 1967.

[6] T. Chaney and F. Rosenberger, "Anomalous Behavior of Synchronizer and Arbiter Circuits," IEEE Trans. on Computers, vol. C-22, pp. 421-422, April 1973.

[7] James Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, 1981.

[8] J. R. Cox and D. M. Zar, “Synthesis of Control Elements from Petri Net Models,” WU Tech Report, WUCSE-2005-43, Sep. 2005.

[9] Neil Weste, David Harris, *CMOS VLSI Design: A Circuit and Systems Persepctive(3rd Ed.)*, Pearson Education, Inc., pp. 453-463, 2004.