Berger Code-Based Concurrent Error Detection in Asynchronous Burst-Mode Machines

Sobeeh Almukhaizim and Yiorgos Makris
Electrical Engineering Department, Yale University
New Haven, CT 06520, USA

Abstract

We discuss the use of the Berger code for Concurrent Error Detection (CED) in Asynchronous Burst-Mode Machines (ABMMs). We present a state encoding method which guarantees the existence of the two key components for Berger-encoding an ABMM, namely an inverter-free ABMM implementation of the circuit and an ABMM implementation of the corresponding Berger code generator. We also propose improved solutions to two inherent problems of CED in ABMMs, namely checking synchronization and detection of error-induced hazards. Experimental results demonstrate that Berger code-based CED reduces significantly the cost of previous CED methods for ABMMs.

1. Berger-Encoded ABMMs

In a Berger-encoded circuit, single errors lead to a non-codeword as long as they result in transitions in either the $0 \rightarrow 1$ or the $1 \rightarrow 0$ direction at the output, but not both. The authors in [1] show how to enforce this constraint by redesigning a combinational circuit such that inverters only appear at the inputs, which guarantees that all single errors will only cause unidirectional error effects at the outputs. In contrast to the circuits targeted in [1], ABMMs have combinational feedback and impose additional constraints in order to guarantee the existence of a hazard-free implementation [2]. We demonstrate next how to obtain the *two key components* necessary for designing a Berger-encoded ABMM: i) an inverter-free ABMM implementation, and ii) an ABMM implementation of the Berger code generator.

In order to generate an inverter-free ABMM implementation, we propose to re-encode the states of the ABMM such that states are uniquely distinguished without the need for inverted state bits. MINIMALIST, a comprehensive synthesis tool for ABMMs, uses *dichotomies* [3] to represent the constraints that must be satisfied by the state codes for an ABMM implementation to exist. A dichotomy, such as ($Group_1$;$Group_2$), specifies two groups of incompatible states and a solution assigns a logic value, x, to a bit in the state encoding of all the states in one group, say $Group_1$, while assigning $\overline{x}$ to the same bit in all the states in the other group, $Group_2$. MINIMALIST encodes the states such that every state is distinguished from incompatible states through a positive (negative) identifier; i.e. a bit that is assigned a logic '1' (logic '0') to solve the dichotomy, respectively. In order to eliminate inverters, we augment the state encoding with the opposite value of a state bit if the corresponding state bit is used as a negative identifier to solve some dichotomy. The new bit now becomes a positive identifier for the same dichotomy, eliminating the need for inverters.

In order to derive the specification of the Berger code generator, we start from the specifications of the original ABMM and we substitute the output burst with the corresponding Berger code. Subsequently, MINIMALIST is used to synthesize the ABMM implementation of the Berger code generator.

2. Berger Code-Based CED for ABMMs

In order to perform Berger code-based CED for ABMMs, two additional challenges need to be addressed [4]: identification of appropriate checking time (checking synchronization) and detection of errors that only cause hazards (detection of error-induced hazards). Methods to address both of these problems were described in [4]. In this section, we propose improved solutions, which reduce the overhead and enhance the efficiency, and we demonstrate the complete Berger code-based CED method for ABMMs.

Checking Synchronization: Based on the the definition of ABMMs, the output of the Berger-encoded ABMM is steady during an input burst and only changes after the input burst is complete. When the input burst is complete, however, the ABMM and the Berger code generator operate asynchronously and their result is only guaranteed to be stable when the first bit of the new input burst arrives. Thus, the Berger code checker should be activated (deactivated) during (in between) input bursts. In order to control the result of the checker, we use a Transition Prediction Function (*TPF*) which signifies the end of an input burst. Since the TPF controls the checking, its implementation should be hazard-free to ensure that no false positives or false negatives occur. Therefore, the TPF is also implemented as an ABMM. The TPF obtains a logic value of '1' when the current input burst and state combination results in a state transition and/or a change at an output, and a logic value of '0' otherwise.

3-9810801-0-6/DATE06 © 2006 EDAA

Circuit		Original		Inverter-Free		Code Generator			T.P.F.		Checker [5]		H. D. Circuit		Synchronizer		Total	
Name	I/S(Bits)/O	Lit.	Gates	Lit.	Gates	k	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates	Lit.	Gates
concur-mixer	3/3(4)/3	26	16	50	28	1	22	13	29	17	4	2	22	12	15	8	150	84
martin-q-element	2/2(2)/2	14	9	20	11	1	3	1	1	0	3	2	14	8	10	5	59	31
opt-token-distributor	4/6(6)/4	74	41	115	61	2	32	18	1	0	5	3	29	16	20	11	210	113
pe-send-ifc	5/5(5)/3	110	58	153	79	2	128	67	134	79	5	3	22	12	15	8	465	252
tangram-mixer	3/2(2)/2	17	10	24	15	2	17	10	1	0	4	2	14	8	10	5	78	44
p1	13/11(8)/14	458	238	615	320	3	200	108	135	76	11	5	104	59	70	41	1143	613
while_concur	4/4(4)/3	41	24	59	33	1	19	11	31	18	4	2	22	12	15	8	158	88
rf-control	6/6(6)/5	75	37	114	63	1	24	14	29	17	5	3	37	21	25	14	242	136
hp-ir	3/2(2)/2	13	8	23	14	1	11	5	33	19	3	2	14	8	10	5	102	57
while	4/3(4)/3	27	16	47	27	1	5	2	1	0	4	2	22	12	15	8	102	55

Table 1. Experimental Results for Berger Code-Based CED

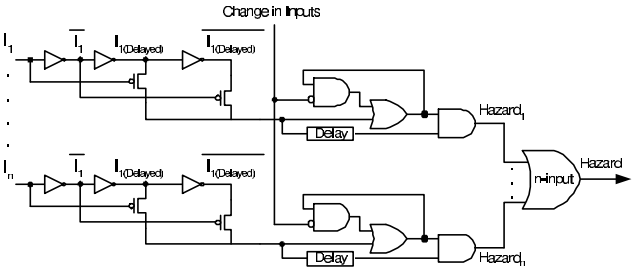

Figure 1. Hazard Detection Circuit

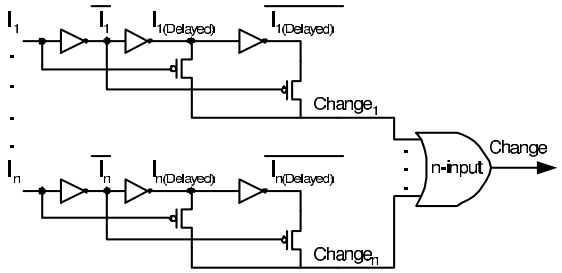

Figure 2. Change Detection Circuit

Detection of Error-Induced Hazards: In order to detect error-induced hazards, we designed the circuit illustrated in Fig. 1, which monitors the number of transitions that occur on an output. Hazard detection is based on the fact that no more than one transition, either rising or falling, is allowed on an output after every input change. Therefore, the change detection circuit of Fig. 2 is added to reset the value of feedback loop signal whenever an input changes. Subsequently, the feedback loop in Fig. 1 monitors the output and latches a logic '1' value if it makes a transition. If a second transition occurs on the same output bit, then the hazard signal is asserted.

Complete Berger Code-Based CED Method: The proposed CED method is illustrated in Fig. 3. First, the ABMM is Berger-encoded as described in Section 2. Then, the ABMM implementation of the TPF is added to enable/disable the checker based on the aforementioned checking synchronization method. Subsequently, the hazard detection circuit, including the change detection circuit, is added to detect error-induced hazards. Finally, a few glue-logic gates (G1-G3) are used to generate the error output.

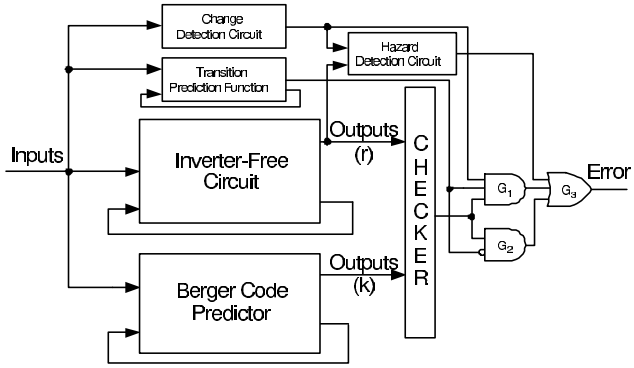

Figure 3. Berger Code-Based CED

3. Experimental Results & Conclusions

The results for Berger code-based CED are summarized in Table 1. For a few circuits, the cost of the TPF is zero. This is attributed to the simplicity of the specification of these controllers, wherein every input burst is composed of a single input change and, hence, the TPF always indicates a transition. On average, the inverter-free ABMM implementation increases the area cost by 50% over the area cost of the original ABMM implementation. Compared to the CED methods in [4], Berger code-based CED reduces the area cost, on average, by more than 16%.

References

[1] N. K. Jha and S.-J. Wang, "Design and synthesis of self-checking VLSI circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 6, pp. 878–887, 1993.

[2] R. M. Fuhrer and S. M. Nowick, *Sequential Optimization of Asynchronous and Synchronous Finite-State Machines: Algorithms and Tools*, Kluwer Academic Publishers, 2001.

[3] G. De Micheli, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Optimal state assignment for finite state machine," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 4, no. 3, pp. 269–285, 1985.

[4] S. Almukhaizim and Y. Makris, "Concurrent error detection in asynchronous burst-mode controllers," in *Design Automation and Test in Europe Conference*, 2005, pp. 1272–1277.

[5] X. Kavousianos and D. Nikolos, "Novel single and double output TSC berger code checkers," in *VLSI Test Symposium*, 1998, pp. 348–353.